

[Loading MPS to PDF converter (version 2006.09.02).]



Meaning and Understanding in Human-centric AI

Narrative-based Understanding of Everyday Activities: A Cookbook

Editors:

Luc Steels (VIU)
Robert Porzel (UHB)

Contributors:

Anna Morbiato (VIU)
Patrizia Cani (CaV)
Luc Steels (VIU)
Nikolaos Kondylidis (VUA)
Annette Ten Teije (VUA)
Ilaria Tiddi (VUA)
Remi van Trijp (CSL)
Katrien Beuls (UNamur)
Paul Van Eecke (VUB)
Jens Nevens (VUB)
Robin De Haes (VUB)
Jonas Doumen (VUB)
Lara Verheyen (VUB)
Laura Spillner (UHB)
Vanja Cangalovic (UHB)
Rachel Ringe (UHB)
Mihai Pomarlan (UHB)
Nima Zargham (UHB)
Rainer Malaka (UHB)
Robin Nolte (UHB)
Robert Porzel (UHB)

Edited by Luc Steels (Venice International University) and Robert Porzel (University of Bremen)

Cover Graphic Design - studio +fortuna, Trieste

Published by Venice International University, Isola di San Servolo 30133 Venice, Italy

Open Access: <https://zenodo.org/communities/muhai/>

Citation:

Steels, L. and Porzel, R. (eds.) Narrative-based Understanding of Everyday Activities: A Cookbook. Venice International University, Venice.

DOI: 10.5281/zenodo.6602456

Volume published in the framework of the MUHAI project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 951846. It reflects only the authors' view and the Commission is not responsible for any use that may be made of the information it contains.

This publication is released under the Creative Commons Attribution 4.0 International license, providing Open Access through zenodo.org. Copyrights for all articles and figures are retained by their authors or copyright holders. You can freely share, adapt and draw on this work as long as you give credit, as per the terms of the license. If you reproduce or draw on material from this publication, we'd be grateful if you could give credit and link back. The editors have used their best endeavours to ensure that the URLs for external websites referred to in this book are correct and active at the time of going to press. However, the editors have no responsibility for the websites and can make no guarantee that a site will remain live or that the content is or will remain appropriate. Every effort has been made to trace all copyright holders, but if any have been inadvertently overlooked the editors will be pleased to include any necessary credits in any subsequent reissue or edition.



This project has received funding from the European Pathfinder Project under Grant Agreement N° 951846

Executive Summary

MUHAI is an EU project that studies how it is possible to build AI systems that rest on meaning and understanding. We want to go beyond *cognition* based on the use of statistically acquired pattern recognition and pattern completion and integrate *sapience* based on structured knowledge representations, reasoning and constructivist learning.

This book is the second open access volume in a series reporting results of the project. The first volume, published as Steels, L., ed. (2022) Foundations for Meaning and Understanding in Human-centric AI. Venice International University, Venice. (<https://zenodo.org/record/6666820>), surveyed how the notion of understanding is being discussed and treated in other human-centred research fields, more specifically in social brain science, social psychology, linguistics, semiotics, economics, social history and medicine. The main conclusion was that understanding is the process of constructing a coherent narrative about a situation that integrates and is compatible with a multitude of knowledge sources. The knowledge sources range from visual and motor experiences to language processing, semantic memory, and episodic memory.

In this second volume the challenge of meaning and understanding is explored with concrete case studies that are all about understanding everyday activities in the real world. The case studies come from the domain of cooking: the preparation and execution of recipes and the invention of recipes by constructing variations of existing ones. The different papers push the state of the art in technologies needed to operationalize understanding: computational linguistics, semantic web technology, knowledge-based robotics, and generative AI. There are also contributions on data structures and strategies for integrating these components in the construction of integrative narrative networks.

The volume is organized into a series of chapters:

- Chapter 1. entitled 'Introduction' introduces the philosophical and conceptual framework that underlies the general approach taken to meaning and understanding in this volume.
- Chapter 2. entitled 'Benchmarks' introduces concrete challenges that can help to push forward the state of the art and also reports on the first experiments to achieve these benchmarks. There are two benchmarks: (i) Understanding a recipe in the sense of being able to deconstruct the primitive actions that are needed to prepare the instructions, despite incompleteness and ambiguity, and (ii) Understanding how people perform ingredient substitution in recipes by interactively learning how ingredients and ingredient properties affect such decisions.
- Chapter 3. entitled 'Semantic and Pragmatic Foundations' is oriented towards linguistics as a source of insights and techniques for understanding better how language provides information through lexical and grammatical structures for reconstructing meaning from text.
- Chapter 4. entitled 'Computational Tools' introduces generic building blocks for implementing meaning and understanding in AI systems. The tools range from construction grammars for precision-parsing to representation languages for capturing the semantics of phrases and the larger-scale structure of multi-phrase narratives.
- Chapter 5. entitled 'Interaction' focuses on a crucial feature of human-centred AI, namely the interaction with a human user. It raises issues with respect to achieving explainability and trust.

Contents

I	Introduction	2
1	Decomposing Mental Complexity	4
2	Serving an Ontological Model for Narratives	11
II	Benchmarks	20
3	A Benchmark for Recipe Understanding	22
4	Handling the Cooking Benchmark	41
5	A Cooking Benchmark for Ingredient Substitution	60
III	Semantic and Pragmatic Foundations	76
6	The Linguistic Ingredients of Recipes	78
7	Pragmatics as the Secret Ingredient	101
8	An Ontological Representation of Imperative Meaning	124
IV	Computational Tools	138
9	Cutting up Recipes	140
10	Integrative Narrative Networks	154
11	Learning large-scale construction grammars	180
V	Interaction	195
12	Conceptual Shadows	197
13	Understanding Uncommon Ground	212
14	Measuring Trust	226

Part I

Introduction

Although the foundations for Meaning and Understanding in Human-centric AI systems have already been discussed in an earlier volume [5], this chapter briefly recapitulates the distinction between the data-driven syntactic pattern recognition approach to AI, proposed for achieving cognition, and the knowledge-driven semantic approach to AI, associated with sapience. We also discuss the notion of narratives, central to sapient AI.

The first paper in this chapter entitled 'Multiplicity of Minds' by Luc Steels argues that (human) mind is not a single entity but decomposes into a multiplicity of minds: a bodily mind achieved by the body and its ability to adapt and remember, a sentient mind sustaining emotions by a complex hormonal system, a cognitive mind that is able to sustain reactive, goal-directed action by a neuronal system, a sapient mind working on the basis of symbols and symbolic inference, and a conscious mind operating on a meta-level and coordinating and influencing all the other minds, including itself. The neural, machine learning approach to AI, as used in behaviour-based robotics or generative AI, is an effective operational model for the cognitive mind whereas the knowledge-driven methods used in expert decision systems or semantic web applications is an effective model for the sapient mind. The future is in understanding the tight interaction between cognition and sapience, but also to integrate the embodied mind, particularly for areas such as understanding everyday activities. The integration of sentience and cognition remain challenges for the future.

The second paper zooms in on the concept of a narrative, which was shown in Vol 1 to be a core concept in the study of meaning and understanding. The paper is entitled 'An Ontology for Narratives' by Robert Porzel and uses the logical calculus of OWL-DL for representing narratives. The paper also presents the ontological commitments and provides an underlying foundational framework together with the ontological design patterns, relevant to this work.

Chapter 1

Decomposing Mental Complexity

Luc Steels*

*Venice International University

Abstract

The (human) mind is an extraordinarily complex system of which current AI covers only a limited number of aspects. There are two ways in which this complex system can be decomposed. First in terms of functions and sub-functions, such as visual perception, visual interpretation, language analysis, planning, memory, etc. Second, in terms of different mental 'modes', respectively achieving embodiment, sentience, cognition, sapience, and consciousness. For a particular task, such as understanding a text with instructions and executing these instructions, a multitude of functions needs to be recruited dynamically and these functions often evoke processing in different modes.

Keywords: Natural language understanding, mind multiplicity, generative AI, cognition, sapience, embodiment.

1.1 Functional decomposition

1.1.1 The arrival of generative AI.

Generative AI, as incarnated by ChatGPT and similar systems, is rolling like a tsunami over the world, carried by a cadre of highly talented and competent scientists and engineers and a group of audacious venture capitalists and entrepreneurs who use the full power of the Internet to disseminate this kind of AI to the world at an unprecedented speed.

Generative AI is based on statistical inference. It hypothesises that the future will be like the past and therefore that we can compute plausible continuations of the future by observing and reusing patterns found in data from the past. For example, we could observe patterns in a set of texts, for example a large collection of cooking recipes, and use them to construct a new text (a new recipe) that looks similar to previous ones. The patterns do not need to be fully detailed. In fact, the real power of statistical inference comes from using abstract categories that are true for a class of objects and relations that hold between classes of objects. For example, if a recipe says 'cook the water', it does not matter whether water is salted or not, what amount of water is involved, whether the water is in a pot or a pan, or whether it is going to be used for preparing pasta or potatoes. The instructions for cooking the water will be more or less the same regardless.

The definition and analysis of statistical inference goes back to the 18th century. What is new is that AI (more specifically a sub-field of AI known as Machine Learning) has invented various algorithms and made them operational so that this method is now usable on a large scale, not just for science and engineering but also for daily life and common sense problem solving.

The most popular methods at the moment, and the ones used in generative AI, encode patterns as layered weighted ‘neural’ networks. Input and output of these networks take the form of data instances consisting of a data vector, i.e. a sequence of numerical values. The values propagate in the network and transform the input vector progressively into an output vector.

The networks are trained by cycling through huge amounts of data for a particular domain and when deemed to be sufficiently effective on a set of test data, which was not part of the original training data, the network is considered ready for use in open real-world applications. A concrete example of a learning algorithm is known as deep learning. In each step of the training phase, part of the vector for a specific data point from the training set is occluded and the network is asked to complete it. When a wrong completion is produced, the weights in the network are adjusted to make it more likely that next time around the output of the network will be closer to the original data point. The adjustment is not random but based on computing the gradient that leads from the erroneous output to the original. Hence the name ‘gradient descent’ for this class of machine learning algorithms. Gradient descent is iterated until the error, in other words the difference between a predicted completion of a partly occluded data vector and the original data vector is negligible for all of the data vectors available to train the network. Needless to say that this process takes time, a lot of time, depending on the size of the network and the amount of training data to be processed.

It may seem unbelievable that a very simple method such as gradient descent is able to acquire the appropriate weights for a network, particularly if the network has millions, even billions, of weights and uses not just two but many layers. Admittedly there are mathematical proofs demonstrating that the network will eventually settle - given a finite amount of training data and enough time to cycle through the data. Nevertheless a theoretical possibility does not yet mean a practical reality.

It turns out however that today we have enough massive computing power and enough data to scale statistical inference up sufficiently to be usable on real world problems. By the early 2020s the statistical inference paradigm had been scaled up to such an unimaginable extent that networks began to appear that were trained with so much cross-domain data that they began to show characteristics of Artificial General Intelligence. They were no longer specific to a particular domain or task but would perform across domains and tasks, and even to tasks for which they were not specifically trained for. What we now see is the birth of a new industry and a gold rush where companies compete to become leader in the field or at least play a significant role.

1.1.2 The Deductive method

These remarkable achievements should not make us forget however that statistical inference is only one of the methods that humans know and use for knowledge acquisition and decision-making. Another method which is often used as well is the deductive method. Deduction is based on (symbolic) domain models containing general facts about a domain. It also relies on strongly regulated inference rules that preserve the truth in applying these general facts for the inference of new facts from the specific facts describing a problem situation. The foundations of the deductive method have been developed over centuries in terms of formal logic, which has been made operational by AI research from the nineteen sixties and seventies to form the basis

of logic programming languages and theorem provers.

The knowledge graphs used today in semantic web technologies are examples of formalized domain models. The design and implementation of these knowledge graphs has also been scaled up to make their application in AI systems such as search engines or semantic web applications feasible.[1] Moreover, various symbolic machine learning methods have been developed to derive ontologies and axioms through abstraction and generalisation or through a constructivist process also known as explanation-based learning.

Today the statistical inference method and the deductive/constructivist method are increasingly used in tandem to create a form of hybrid AI that combines the advantages of both methods. It is also increasingly recognized that the two methods are the foundation of two different types of human thinking, characterised as cognition and sapience, also called system 1 and system 2.

1.2 Cognition and Sapience

Cognition orchestrates behaviors in order to achieve goals and reacts in an adaptive and timely manner to stimuli. It does this by categorizing situations and invoking a response that has been proven to be effective in the past. Cognition is the province of the *cognitive mind*. It works automatically and swiftly. You are walking on the street and an old friend is passing by. Your cognitive mind recognizes her instantly, even if you have not seen her for many years. And you react immediately to greet her warmly. Here is another example. If you are driving your bicycle, your cognitive mind orchestrates the rhythmic pushing of the pedals, or reacts quickly to avoid obstacles such as a dog suddenly crossing the street. All this happens spontaneously and effortlessly.

To properly describe the role of the cognitive mind and predict its behavioral effects, a goal-directed or *teleological stance* is the most appropriate. You have to view the organism as an agent that has the ability to establish goals and subgoals and to enact those behaviors that best satisfy these goals. Goals include reacting timely and appropriately to the environment, but also finding food when hungry or tying shoelaces before going out.

Sapience is also about setting and satisfying goals. But the *sapient mind* goes a step further than the cognitive mind in terms of sophistication because it builds conceptual models to compare different options and choose the best course of action. The sapient mind engages in logical and associative inference and builds narrative structures to make sense of experiences. The cognitive mind is reactive, the sapient mind deliberative. The sapient mind dominates when your attention and energy is entirely focused on ‘intellectual’ work: writing an essay, working out a physics problem, performing calculations, figuring out why a technical device does not work, planning a complex trip, engaging in a deep conversation.

To predict or explain the role of the sapient mind, it is most effective to adopt a *knowledge-level stance*: You assume that a sapient mind has declarative knowledge about the world and uses that to make models in the form of narratives. These narratives help to make sense of the past or form expectations about the future [4]. Moreover to understand why a particular goal has been chosen by the sapient mind, you need to adopt an *intentional stance*, which describes and predicts decisions in terms of beliefs, desires and intentions [2].

The distinction between cognition and sapience is not always made, but there are many reasons for doing so.¹ You have to take into account differences in scope, depth, utility, reliability and

¹In fact the situation is even more complex because the word cognition was for a while exclusively used for the sapient mind.

reaction time of decision processes. These differences have been well documented by psychologist Daniel Kahneman in his book ‘Thinking Fast and Slow’ [3]. He uses the term system 1 for the cognitive mind and system 2 for the sapient mind.

According to Kahneman, System 1 (the cognitive mind) achieves reactive intelligence. It is fast, provides direct solutions, operates involuntarily and seemingly does not require any effort. Fast reactivity does not imply that System 1 cannot anticipate, in fact it does most of the time because anticipation is an important strategy for reacting timely and appropriately to stimuli. System 1 is at work when you instantly recognize your mother in an old family picture, go through the motions of making coffee in your own kitchen without thinking about it, complete an often heard phrase, or answer a question that you have already answered hundreds of times. Importantly, System 1, by itself, cannot explain how it came to a solution. The tennisplayer cannot explain how she hits the ball flying towards her at 120 km/hour.

In contrast, System 2 (the sapient mind) achieves deliberative intelligence. It is slow and takes time to arrive at a solution. It operates under voluntary control, and feels like requiring effort and concentration. Now it is possible to come up with an explanation and formulate it in human natural language or to argue why you did things in a particular way. Kahneman and his collaborators have shown abundantly that System 2 guards the sometimes error-prone and erratic behavior of System 1 by going deeper and taking more time to come up with a conclusion. System 1 tends to use quick heuristics that immediately produce an answer but on second thought are erroneous and shallow.

Clearly, System 1, the cognitive mind, is most appropriate for real-time action and it handles background activities so that your sapient mind can concentrate on something else. In many daily life cases, System 2, the sapient mind, would be too slow to rely on for real-time action and decision-making. On the other hand, when a situation is novel or unexpected, the cognitive mind is at a loss. No solution pops up, or worse, the system comes up with a wrong answer. The sapient mind is therefore appropriate, if not to say crucial, for these more challenging cases, when there is no easy ready-made solution or prior experience, when we have to convince somebody else about a particular solution to a problem, or when the problem is so complex that it cannot be solved by pattern recognition alone.

The general opinion at the moment is that the cognitive mind maps nicely to the neural network mechanisms that implement statistical inference, whereas the sapient mind maps to the deductive rational machinery that has been used in symbolic AI based on representations in a variant of the predicate calculus and forms of logical inference. In real life the two minds form a seamless team and future trustworthy AI will undoubtedly move further in that direction.

1.3 Embodiment, Sentience and Consciousness

Cognition and sapience are not the only mental modes. There are three more: embodiment, sentience, and consciousness (see Figure 1.1)

Embodiment is about the capacities and activities of the body as a material living system. To properly describe the physical behaviour of the body we have to take a *physical stance* and a *functional stance*. The physical stance means that we describe the body in terms of its physical features (anatomy, weight, size, length, biochemical composition, genetics, etc.) and use natural physical laws and biochemical processes to explain and predict how these features evolve in time and have an impact in the world. For example, when you fall down from a balcony you are reduced to a physical object undergoing the force of gravity and an observer can predict where you will fall and at what speed, using Newton’s laws of motion. Whatever

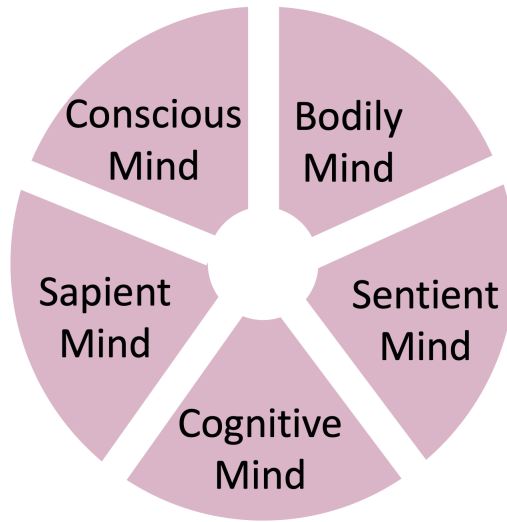


Figure 1.1: The paper proposes a multiplicity of minds that cooperate to achieve global coherent behaviour and decision-making.

goes through your other minds is not going to make any difference.

Here is another example: When you are walking, you spontaneously swing your arms synchronised with the motion of the legs. The resulting pendulous motion helps to create more stability and aids forward movement. The physics of this bodily movement thus helps with locomotion. You are unconsciously exploiting this effect without knowing why. Aspiring roboticists often build so called 2-legged passive walkers as an exercise. These are robots which have no control circuitry to instigate forward movement. They move forward from an inclined plane, using gravity as their only force and exploiting a mechanical design with swinging movements to support their steps and retain balance.²

The body is sometimes said to have a mind of its own. To capture that insight, we can use the term the *bodily mind*. It refers to the body as a physical, living dynamical system that autonomously changes and adapts on the basis of physical and biological principles. For example, when certain movements are made frequently, muscles and metabolic processes adapt to make them more efficient for this task. The body therefore has a kind of memory of the past and it anticipates the future.

The importance of taking the body as having a dynamics of its own which cannot (and should not) be fully controlled by cognition or sapience was the main rallying cry of the behaviour-based robotics movement of the 90s. [6] It is extremely relevant for the study of physically embodied everyday activities which have to exploit real world dynamics of the body and objects in the environment rather than fully control them.

Sentience is primarily concerned with behavioural regulation in reaction to internal or external stimuli, including invasion by foreign bodies. To describe the role of sentience requires an *affective stance*: You can best describe, explain or predict behaviour in terms of emotional states, moods, temperaments and bodily or behavioural reactions to these affective states.

Sentience is enacted by the *sentient mind*. Through the hormonal and immune system it can quickly impact the whole body and cause it to react to sensory stimuli or anticipate positive or negative experiences that may impact the survival or well being. A typical example is the feeling of fear which triggers increased heart rate and blood pressure, faster breathing, tense muscles,

²<https://www.youtube.com/watch?v=FfKQSUhYjIY>

enlargement of the eye pupils (the dark centre of the eye), and changes to the temperature of the hands, making them either more sweaty or more clammy.

Sometimes sentience entirely dominates your existence. You get so angry that rational thought fades away. The body is shaking, your heart is racing, you get white in the face, and you may commit acts you will regret for the rest of your life. Alternatively, when you fall in love, you may start to feel weak in your knees and sense butterflies in your stomach. You want to maximise time with the person you love, make irrational decisions, forget mundane practical matters, lose your self.

At the moment sentience falls almost entirely out of the province of what AI models and systems try to deal with. When AI studies are made of emotion, they mostly concern the recognition of emotional states taken from a short list of prior categories, in other words they focus on the cognitive side of emotion not on emotion itself. But affective states to play a very significant role in all human activities, including cooking. Given that so little has been done in this area, sentience is not further explored here.

Consciousness is considered the highest form of mental activity, enacted by a *conscious mind*. Consciousness operates at a meta-level compared to the other minds. Its role can be viewed as that of the operating system of a (distributed) computer system. It has partial access to information from other minds, seeks to establish a coherent interpretation of what is going on, and influences the other minds by broadcasting information back or on the contrary preventing the flow of information. When information becomes accessible to our conscious mind we say that we have become aware of it. Awareness implies not only sensitivity to signals from the other minds but also being able to interpret these signals and do something with them.

Describing and understanding what a conscious mind does requires a *reflective stance*. You assume that an organism with a conscious mind can reflect on the internal states of all the other mental processes described earlier: embodiment, sentience, cognition, sapience and even consciousness itself, and that this reflection can then impact how each of these minds operates. How this works out in detail is one of the big challenges that a theory of consciousness must reveal but falls outside the scope of the present project, it is however tackled in current projects in the 'Awareness Inside' Portfolio of the EU Pathfinder initiative.³

³<https://eic-emerge.eu/about/eic-pathfinder-challenge>

Chapter 1: References

- [AA1] G. Antoniou and F. Van Harmelen. *A semantic Web Primer*. The MIT Presss, Cambridge Ma, 2008.
- [AA2] Daniel C. Dennett. *The Intentional Stance*. The MIT Press, Cambridge, MA, 1987.
- [AA3] D. Kahneman. *Thinking fast and slow*. Farrar, Strauss and Giroux, New York, 2011.
- [AA4] Allen Newell. The knowledge level. *Artificial Intelligence*, 18(1):81–132, 1982.
- [AA5] Luc Steels. Conceptual foundations for human-centric AI. In Mohamed Chetouani, Virginia Dignum, Paul Lukowicz, and Carles Sierra, editors, *Human-Centered Artificial Intelligence (ACAI 2021)*, pages 8–35, Cham, Switzerland, 2023. Springer.
- [AA6] Luc Steels and Rodney Brooks. *The artificial life route to artificial intelligence: Building embodied, situated agents*. Routledge, 2018.

Chapter 2

Serving an Ontological Model for Narratives

Robert Porzel*

*University of Bremen

Abstract

The activities of people as well as of artificial agents in reality, virtual reality or simulation can be recorded as data that discretize trajectories of body parts and the ensuing force events. While these data provide vast amounts of information they are, by themselves, meaningless. Only when we put them into context we assign a specific meaning to these data. Increasingly, the notion of *narratives* is being used to describe the result of this semiotic process, i.e. we observe events and fit them into a story that makes sense to us. In this work a formal model is presented and discussed that can be employed to represent a narrative using the subset of FOL that is expressible in OWL DL.

Keywords Narratives, Ontologies, Cognitive Systems, Framing

2.1 Introduction

The concept of a *narrative* has migrated from its original domain in the literary sciences to a multitude of diverse and increasingly distant research fields. It has become an important element in research on computer games [7] or in history [9] to name a few of these domains. At long last it has also arrived in the cognitive sciences where narratives are regarded to be a central means of sense making [10]. From there it was merely a short jump over to the field of cognitive robotics where the concept is employed to describe semantically annotated episodes of recorded activities [1].

When we observe people or artificial agents performing everyday activities in reality, virtual reality or simulation, we collect episodic data that represent trajectories of body parts and activity-specific force events. While these data contain large quantities of information they are, by themselves, meaningless. Only when we put them into a pragmatic context we assign specific meanings to these data. For example, we can interpret the same observed episode as either *throwing something* or *dropping something*. This difference in the narrativization, consequently, yields two distinct narratives:

- (1) He dropped the glass onto the floor

(2) He threw the glass onto the floor

Please note that the situation spawning these two minimal narratives can be identical. We will, therefore in our model need to differentiate between a situation, which has not been narrativized and a description of a situation, which pairs a situation with a selected conceptualization, i.e. interpretation, thereof. In addition to becoming meaningful, this pairing will, in turn, evoke a pragmatic stance that ascribes, for example, a specific perspective and intention to the agent(s) acting in the narrative. In the following, we will present a formal theory of narratives using the logical calculus of OWL-DL. Before introducing the theory, we will present the ontological commitments and underlying foundational framework together with the ontological design patterns, relevant to this work.

2.2 Foundational Framework and Prior Art

The model proposed and outlined herein neither stands alone nor is it the first attempt to cast narratives in a formal framework. The following will therefore, briefly describe previous work on formalizing narratives before discussing the foundational commitments and ontological framework employed in this approach.

2.2.1 Related Work

As it should be, each individual approach for representing narratives formally is driven by the specific requirements of the given application at hand. For example, the model of Meghini et al. is seeking to organize information provided in digital libraries and, therefore, equates a narrative with an event and allows for events to feature dependent events [5]. The model uses RDF based on OWL and narrative events can have spatial or temporal relations between them, but the main purpose is to connect digitally represented entities to pertinent events, e.g. the *Divine Comedy* as a book and the person Dante Aligheri can be connected by a narrative *Dante writes the Divine Comedy*. In this approach *events* are also not formally specified as no foundational framework is employed.

A different approach by Kroll et al makes a useful distinction between factual relations, as expresses by knowledge graphs, and narrative relations that constitute hypothetical relation connecting factual ones [11]. For this the approach needs to employ RDF* to express relations that range over relations. This approach can, therefore, be employed to postulate, for example, causation relation as a narrative that connects hitherto isolated knowledge graphs.

A little closer to the task at hand in this work is the work described by Evans et al. that seeks to classify (partial) sensory data as a narrative that can be framed as an inductive logic programming task [6]. While their focus lies on reducing the search space by finding hypotheses – which equal narratives in their approach – that provide as simple an explanations of the observed data as possible. This approach can, therefore, be deemed compatible yet orthogonal to the one presented herein, as it focuses not on representing narratives in an ontology, but on a classification approach that employs such a model as a target representation.

2.2.2 Foundational Commitments

It has become a sensible and important part of ontology engineering to make the underlying foundational commitments of a given model explicit. This formal theory of narratives *cum* model is based on the DOLCE+DnS Ultralite (DUL) foundational framework [8, 26]. This

decision is greatly motivated by the underlying ontological commitments of DUL, its axiomatization as well as the incorporation of the *Descriptions and Situations* module. Firstly, DUL is not a revisionary model, but seeks to express stands that shape human cognition. Furthermore, it assumes a multiplicative approach – however rather than capturing the flexibility of our usage of objects via multiple inheritance it is also possible to combine a reduced *ground* classification with a *descriptive* approach for handling this flexibility. For this, a primary branch of the ontology represents the ground **physical model**, e.g. objects and events, while a secondary branch represents the **social model**, e.g. roles and tasks. All entities in the social branch would not exist without a cognitive agent, i.e. they constitute social objects that represent concepts about or descriptions of entities, as, for example, the construal of an observable event where an object moves from an agent’s hand to the floor into the interpretations given in Examples 1 and 2.

Every axiomatization in the physical branch can, therefore, be regarded as expressing some physical context whereas axiomatizations in the descriptive social branch are used to express social contexts. A set of dedicated relations is provided that connect both branches. For example, the relation *classifies* connects ground objects, e.g. a hammer, with the roles they can play, i.e. potential classifications. Thus, we can state that a hammer can in some context be conceptualized as a murder weapon, a paper weight or a door stopper. Nevertheless, neither its ground ontological classification as a tool will change nor will hammers be subsumed as kinds of door stoppers, paper weights or weapons via multiple inheritance.

2.2.3 The SOMA Ontology

The approach presented herein extends the Socio-physical Model of Activities (SOMA) by including a new module for representing narratives. Naturally, SOMA is also based on the DUL foundational framework and its plugin IOLite [4]. Consequently, SOMA has two knowledge branches; one physical and a social, which leads to a distinction between objects and events in the physical branch on the one hand, as well as roles and tasks in the social branch on the other. Beßler et al. explain that axiomatizations in the physical branch express physical contexts which can be classified by axiomatization in the social context. For example, a glass and its properties of being a designed physical artifact would be described using parts of the physical branch, but its potential usage or affordances would be axiomatized using the social branch [3]. SOMA is built out of multiple modules for different aspects. For example, the *SAY* module which defines the theories required by linguistic processing of instructions [16]. An overview of some of the modules is given in Figure 2.1 including the module for representing narratives (NARR) described in this work.

2.3 A Formal Theory of Narratives

First, we look at the ground partition of the theory, that represents *episodes* that occur in the world and can be recorded, e.g. as visual or force dynamic data.¹ These numeric data can be seen as the type of sensory data used in the approach of Kroll et al (2021).

¹In the following all concepts that are introduced in the SOMA-NARR module will be italicized and concepts already given by the foundational framework will be denoted by the prefix "dul:" that is short for `dul:<http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#>` where further documentation about these concepts can be found.

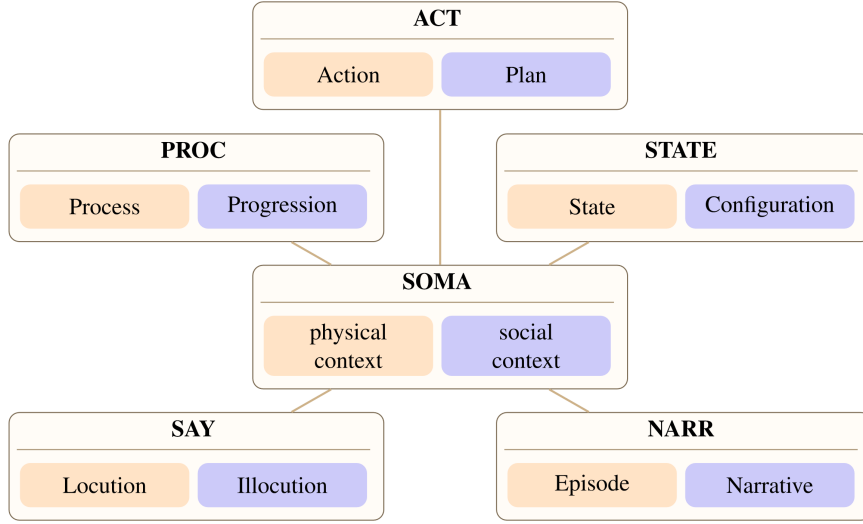


Figure 2.1: An overview of the SOMA modules.

$$\text{Episode}(x) \rightarrow \text{Situation}(x) \quad (2.1)$$

$$\text{Episode}(x) \rightarrow \forall y(\text{INC}_E(y, x) \rightarrow \text{Event}(y)) \quad (2.2)$$

$$\text{Episode}(x) \rightarrow \exists y(\text{INC}_E(y, x)) \quad (2.3)$$

$$\text{Action}(x) \rightarrow \text{Event}(x) \quad (2.4)$$

$$\text{Event}(x) \rightarrow \forall y(\text{HAS}_P(y, x) \rightarrow \text{PhysicalObject}(y)) \quad (2.5)$$

$$\text{Event}(x) \rightarrow \exists y(\text{HAS}_P(y, x)) \quad (2.6)$$

We model an *episode* as a *dul:situation* (2.1) that must include one *dul:event* (please note that an episode that consists of multiple events is not excluded by this axiomatization) denoted by the *includesEvent* (INC_E) relation (2.2,2.3). As already given by the foundational framework *dul:actions* are types of *dul:events* (2.4) and *dul:events* have *dul:physicalObjects* as participants denoted by the *dul:hasParticipant* (HAS_P) relation (2.5,2.6). An *episode* can now be described by exactly one *narrative* (2.7).

$$\text{Episode}(x) \rightarrow \exists y(\text{DESCR}(y, x) \wedge \text{Narrative}(y)) \quad (2.7)$$

$$\text{DESCR}(x, y) \rightarrow \text{Description}(x) \quad (2.8)$$

The *describes* relation (DESCR) holds between a *dul:description* (e.g., an *narrative*) and entities that are conceptualized by the description (2.8).

Narratives define construals of *dul:tasks* and *dul:roles* within the *episode* they describe. Specifically, we distinguish between *dul:tasks* that are conceptualizations of *dul:actions*, and the *dul:roles* that are narrative-specific conceptualizations of the given function of *dul:physicalObjects*. Consequently, we introduce two relations *definesTask* (DEF_T) and *definesRole* (DEF_R) that link a *narrative* to the respective entities.

$$\text{DEF}(x, y) \rightarrow \text{Description}(x) \wedge \text{Concept}(y) \quad (2.9)$$

$$\text{DEF}_T(x, y) \rightarrow \text{DEF}(x, y) \wedge \text{Narrative}(x) \wedge \text{Task}(y) \quad (2.10)$$

$$\text{DEF}_R(x, y) \rightarrow \text{DEF}(x, y) \wedge \text{Narrative}(x) \wedge \text{Role}(y) \quad (2.11)$$

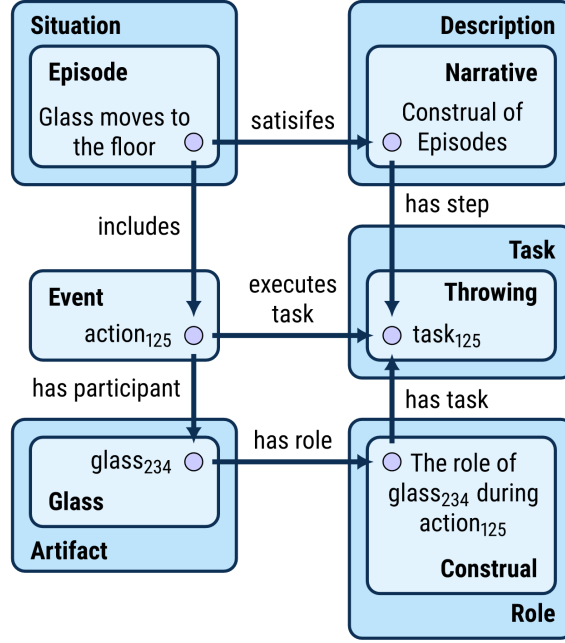


Figure 2.2: An example of an episode (the physical occurrence) and its narrativization.

Finally, we formalize the *narrative* concept by axiomatizing its relationship to the episodic content described, and the concepts defined by it.

$$\text{Narrative}(x) \rightarrow \text{Description}(x) \quad (2.12)$$

$$\text{Narrative}(x) \rightarrow \forall y(\text{DESCR}(x, y) \rightarrow \text{Episode}(y)) \quad (2.13)$$

$$\text{Narrative}(x) \rightarrow \exists y(\text{DEF}_T(x, y) \wedge \text{Task}(y)) \quad (2.14)$$

$$\text{Narrative}(x) \rightarrow \exists y(\text{DEF}_R(x, y) \wedge \text{Role}(y)) \quad (2.15)$$

Our theory views *narratives* as *dul:descriptions* (2.12) that only describe *episodes* (2.13). A *narrative* defines exactly one *dul:task* (2.14), and one *dul:role* (2.15). These relationships between concepts defined in our theory are depicted in Figure 2.2.

$$\text{NAR}(x, y) \rightarrow \text{Episode}(x) \wedge \text{Concept}(y) \quad (2.16)$$

$$\text{NAR}(x, y) \rightarrow \exists a(\text{DESCR}(a, x) \wedge \text{DEF}(a, y)) \quad (2.17)$$

$$\text{NAR}_R(x, y) \rightarrow \text{NAR}(x, y) \wedge \text{Role}(y) \quad (2.18)$$

$$\text{NAR}_R(x, y) \rightarrow \exists a(\text{DESCR}(a, x) \wedge \text{DEF}_R(a, y)) \quad (2.19)$$

$$\text{NAR}_T(x, y) \rightarrow \text{NAR}(x, y) \wedge \text{Task}(y) \quad (2.20)$$

$$\text{NAR}_T(x, y) \rightarrow \exists a(\text{DESCR}(a, x) \wedge \text{DEF}_T(a, y)) \quad (2.21)$$

The *narrativizes* relation links *episodes* and *dul:concepts* (2.16). Any *dul:concept* that is linked to a ground entity in an *episode*, e.g. via the *dul:classifies* relation, is defined in a *narrative* that describes the *episode* (2.17). Specifications of this relation further constrain the type of the construed *dul:concept* (2.18), and how the *dul:concept* is related to the *narrative* that defines it (2.19). Another specifications of this relation further constrain the type of the narrativized *dul:task* (2.20), and how the *dul:concept* is related to the *narrative* that defines it (2.21). The formalization of other sub-relations, i.e., *narrativizesRole* and *narrativizesTask*, is done analogously. The manifestation of an *narrative* (*MNAR*) is, in our view, a *dul:situation* that

satisfies (*SAT*) the *narratives* describing the episodic entities that are included in the *dul:situation* (2.22). More concretely, it is a *dul:situation* where an *dul:agent* executes the *dul:task* defined by the *episode* by following a *dul:plan* (which is another type of *dul:description*) involving *dul:physicalObjects* playing certain *dul:roles* and *dul:regions* setting specific *dul:parameters* for that execution. Hence, *dul:situations* in which *narratives* are manifested also satisfy the *dul:plan* that the *dul:agent* executes (2.23).

$$MNAR(x) \rightarrow \exists!y(SAT(x, y) \rightarrow Narrative(x)) \quad (2.22)$$

$$MNAR(x) \rightarrow \exists!y(SAT(x, y) \rightarrow Plan(x)) \quad (2.23)$$

A formalization of *dul:plans* that describe *dul:tasks* evoked by *episodes* is part of the existing DUL ontology. What makes a narrative a special type of description is, among other things, that it assumes a specific *perspective*. As this need to be included in a corresponding model, we propose to extend the work from spatial cognition to the narrative domain as follows:

$$Perspective(x) \rightarrow Abstract(x) \quad (2.24)$$

$$HASO(x, y) \rightarrow Perspective(x) \wedge Origo(y) \quad (2.25)$$

$$Origo(x) \rightarrow Role(x) \quad (2.26)$$

$$Egocentric(x) \rightarrow Perspective(x) \wedge Origo(Speaker) \quad (2.27)$$

$$Allocentric(x) \rightarrow Perspective(x) \wedge Origo(\neg Speaker) \quad (2.28)$$

Lastly, we can connect *narratives* with their assumed *perspective* by postulation the relation of having a point of view (HAS_{PoV}) that holds between *narratives* and *perspectives* (2.29).

$$Narrative(x) \rightarrow \forall y(HAS_{PoV}(y, x) \rightarrow Perspective(y)) \quad (2.29)$$

2.4 Discussion and Future Work

This work is by no means done and, therefore, still in progress. Along with assuming a specific perspective on an episode, narratives also feature a teleological stance and in many cases also a normative valence. This needs to be included to arrive at a comprehensive model that allows for reasoning about narratives, e.g. what the specific differences between two distinct narrativizations of an identical episode are and even what they mean. This type of reasoning would extend the semantics employed, for example, in opinion mining and sentiment analysis, as narratives could then be grouped and compared in terms of perspective, stance or valence.

The contribution of the work presented herein is to provide a representational framework that can readily be employed in cognitive robotics to counterpart the notion of a *task* that is given to a robotic agent and can be executed by finding an appropriate action with the notion of a *narrative* that looks at an action and seeks to make sense of it. Ideally, one could match the task leading to an action and the narrative describing the action to express if that task has been successfully executed by the agent from the point of view of the narrativizer. As most readers will know there can be vast differences in these judgments, for example, between parents and children concerning the question if a room has been properly cleaned.

Again, it is important to note that this approach explicitly rejects an objective notion of a narrative, i.e. to equate a narrative with what has objectively happened and can be recorded

an stored as data. As a descriptive notion a narrative assumes a specific point of view on the episodic event. It, therefore, provides a *spin* on the event and is subjective. Nevertheless, these views can be shared by collectives and become well established frames in which larger historical or everyday episodic events can be seen by societies or groups.

Next to finishing the model proposed herein and populating it for the domains of everyday activities and social policies as it is planned in the MUHAI project, we currently implementat a social observatory where narrative networks as personal dynamic memories (PDMs) [14] can be detected in large collections of knowledge graphs concerning social equality and policy. In the former project in cognitive robotics narrative enabled episodic memories (NEEMS) [2] are constructed and stored for analysis that describe executions of everyday activities.

Chapter 2: References

- [AB1] Michael Beetz, Daniel Beßler, Andrei Haidu, Mihai Pomarlan, Asil Kaan Bozcuoglu, and Georg Bartels. Knowrob 2.0 – a 2nd generation knowledge processing framework for cognition-enabled robotic agents. In *International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, 2018.
- [AB2] Michael Beetz, Daniel Beßler, Sebastian Koralewski, Pomarlan Mihai, Abhijit Vyas, Alina Hawkin, and Kaviya Dhanabalachandran. NEEM Handbook. <https://ease-crc.github.io/soma/owl/1.1.0/NEEM-Handbook.pdf>, 2020. Accessed: 2021-04-28.
- [AB3] Daniel Beßler, Robert Porzel, Mihai Pomarlan, Michael Beetz, Rainer Malaka, and John Bateman. A Formal Model of Affordances for Flexible Robotic Task Execution. In *European Conference on Artificial Intelligence (ECAI)*, 2020.
- [AB4] Daniel Beßler, Robert Porzel, Mihai Pomarlan, Abhijit Vyas, Sebastian Höffner, Michael Beetz, Rainer Malaka, and John Bateman. Foundations of the Socio-physical Model of Activities (SOMA) for Autonomous Robotic Agents. *CoRR*, 2020.
- [AB5] Valentina Bartalesi Carlo Meghini and Daniele Metilli. Representing narratives in digital libraries: The narrative ontology. *Semantic Web Journal*, 2021.
- [AB6] Richard Evans, José Hernández-Orallo, Johannes Welbl, Pushmeet Kohli, and Marek Sergot. Making sense of sensory input. *Artificial Intelligence*, 293:103438, 2021.
- [AB7] Tracy Fullerton. *Game Design Workshop. A Playcentric Approach to Creating Innovative Games*. 01 2008.
- [AB8] Aldo Gangemi and Peter Mika. Understanding the semantic web through descriptions and situations. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE - OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003*, pages 689–706, 2003.
- [AB9] Yuval N Harari. *Sapiens : a Brief History of Humankind*. Harper, New York, 2015.
- [AB10] David Herman. Narrative theory and the cognitive sciences. *Narrative Inquiry*, 11:1–34, 01 2003.
- [AB11] Hermann Kroll, Denis Nagel, and Wolf-Tilo Balke. Modeling narrative structures in logical overlays on top of knowledge repositories. In Gillian Dobbie, Ulrich Frank, Gerti Kappel, Stephen W. Liddle, and Heinrich C. Mayr, editors, *Conceptual Modeling*, pages 19–33. Springer, Heidelberg, 2020.
- [AB12] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. WonderWeb deliverable D18 ontology library (final). Technical report, IST Project 2001-33052 WonderWeb: Ontology Infrastructure for the Semantic Web, 2003.

- [AB13] Robert Porzel and Vanja Sophie Cangalovic. What Say You: An Ontological Representation of Imperative Meaning for Human-Robot Interaction. In *Proceedings of the JOWO – Ontology Workshops*, Bolzano, Italy, 2020.
- [AB14] Luc Steels. Personal dynamic memories are necessary to deal with meaning and understanding in human-centric AI. In Alessandro Saffiotti, Luciano Serafini, and Paul Lukowicz, editors, *Proceedings of the First International Workshop on New Foundations for Human-Centered AI co-located with 24th European Conference on Artificial Intelligence (ECAI 2020), Santiago de Compostella, Spain, September 4, 2020*, volume 2659 of *CEUR Workshop Proceedings*, pages 11–16. CEUR-WS.org, 2020.

Part II

Benchmarks

This volume explores operational models of the mental processes underlying the understanding of natural language texts that describe everyday activities, with the cooking domain as the primary source of examples and challenges. Understanding can only be measured indirectly, namely by testing the performance on mental tasks that require understanding. Two such tasks are discussed. They have been translated into concrete published benchmarks:

- to come up with a set of actions so that a robot might be able to execute a recipe either in the real world or in a virtual world (section 2.1)
- to construct variations of a recipe in order to adapt it to other real world circumstances, such as avoiding allergies, taking into account available stock, etc. (section 2.3.)

We also present in this chapter concrete solutions to these two benchmarks. For the task execution benchmark, we present a fully implemented system (section 2.2.) that uses symbolic techniques in the form of a construction grammar for analysing the recipe and a constraint-based procedural semantics system for defining and simulating concrete actions. For the ingredient substitution task, we show a hybrid system (section 2.3.) that uses both data-driven AI and knowledge graphs.

Other chapters of this volume will go into greater detail on the technology used here for achieving the two benchmarks.

Chapter 3

A Benchmark for Recipe Understanding in Artificial Agents¹

Jens Nevens*, Robin De Haes*, Rachel Ringe[†], Mihai Pomarlan[‡], Robert Porzel^{†,◊} Katrien Beuls^{△,◊}, Paul Van Eecke^{*,◊}

*Artificial Intelligence Laboratory, Vrije Universiteit Brussel

[†]Digital Media Laboratory, University of Bremen

[‡]Department of Applied Linguistics, University of Bremen

[△]Faculté d’Informatique, Université de Namur

Abstract

This paper introduces a novel benchmark that has been designed as a test bed for evaluating whether artificial agents are able to understand how to perform everyday activities based on a set of instructions, with a focus on the cooking domain. Understanding how to cook recipes is a highly challenging endeavour due to the underspecified and grounded nature of recipe texts, combined with the fact that recipe execution is a knowledge-intensive and precise activity. The benchmark comprises a corpus of recipes, a procedural semantic representation language of cooking actions, qualitative and quantitative kitchen simulators, and a standardised evaluation procedure. Concretely, the benchmark task consists in mapping a recipe formulated in natural language to a set of cooking actions that is precise enough to be executed in the simulated kitchen and yields the desired dish. To overcome the challenges inherent to recipe execution, this mapping process needs to incorporate reasoning over the recipe text, the state of the simulated kitchen environment, common-sense knowledge, knowledge of the cooking domain, and the action space of a virtual or robotic chef. This benchmark thereby addresses the growing interest in human-centric systems that combine natural language processing and situated reasoning to perform everyday activities.

Keywords: benchmark, recipe execution, natural language understanding, situated reasoning

¹This paper has been published as Nevens, J., De Haes, R., Ringe, R., Pomarlan, M., Porzel, R., Beuls, K., Van Eecke, P. (Accepted/In press). A Benchmark for Recipe Understanding in Artificial Agents. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*.

3.1 Introduction

Recipes are a type of procedural text that many people interact with in their daily lives. Because of this familiarity, the cooking domain is often used as a test bed to assess the ability of artificial agents to learn to perform everyday activities [6, 19, 16]. However, having agents understand how to execute recipes is a highly challenging endeavour due to the underspecified and grounded nature of recipe texts and the fact that recipe execution is a knowledge-intensive and precise activity. Indeed, recipe texts are grounded in the world since many co-references and anaphoric expressions in recipes do not have an antecedent in the recipe text itself, but rather refer to the result of the execution of earlier recipe steps. Consequently, such co-references cannot be resolved by reasoning over the recipe text alone, but also require to keep track of the state changes they entail in the kitchen environment. Recipe texts are also highly underspecified as information that seems trivial for humans preparing the dish is often completely left out or incomplete through the use of null arguments and ellipses [19, 28]. However, for artificial agents, this information is crucial in order to carry out the cooking instructions correctly. Therefore, any missing information has to be derived from either common-sense knowledge or knowledge of the cooking domain and the cooking instructions need to be specified a level of precision that allows them to be executed in a simulated kitchen environment.

To illustrate these challenges, consider the following cooking instructions:

Mix together flour, water, and eggs
Roll the dough into small balls
Place on a baking sheet

In the first instruction, the act of cracking the eggs before they can be mixed with flour and water is completely omitted from the recipe text. In this case, an artificial agent would need to combine situated reasoning, common-sense knowledge and knowledge of the cooking domain to derive that (i) eggs can be cracked, (ii) when cracked, eggs consist of egg whites and yolks, (iii) uncracked eggs would lead to egg shells when mixed, (iv) egg shells are not edible, and thus (v) the eggs should be cracked before going in the mixture. Similar examples include the peeling of some vegetables before they can be cut (like onions) or the default tool to use for carrying out certain cooking actions such as beating egg whites (namely a whisk) or cutting onions (namely a knife).

In the second instruction, the noun phrase “*the dough*” does not refer to an entity introduced earlier in the recipe text, but to the result of performing the first cooking action. Thus, to find this referent, the agent has to perform the mixing action and then use knowledge of the cooking domain, e.g. concerning the ingredients of the mixture and the mixture’s consistency, to find out that this mixture can indeed be called dough, as opposed to other food products that are present in the kitchen.

The patient of the placing action in the third instruction is left out, but implicitly refers to the result of the previous cooking action. Here, one has to perform the rolling action and then reason over the recipe text, taking into account the sequential nature of cooking instructions as in [19] and [16], to find that it is the resulting balls of dough that should be placed on a baking sheet.

The benchmark that we propose in this paper aims to address the challenges inherent to recipe execution. In particular, the goal is to assess the ability of an artificial agent to follow a recipe specified in natural language and prepare the dish in a simulated kitchen environment. Concretely, the benchmark comprises (i) a corpus of 30 recipes, (ii) a procedural semantic representation language specifying 38 cooking actions, (iii) qualitative and quantitative kitchen

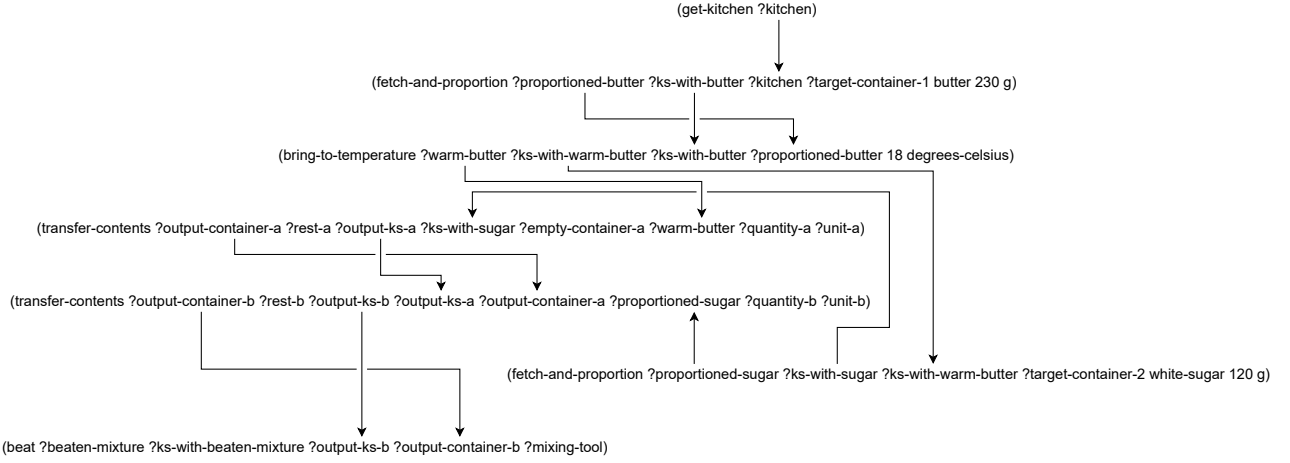


Figure 3.1: A possible set of cooking actions for the cooking instructions “230 grams of butter, room temperature. 120 grams of sugar. Beat the butter and the sugar together.” This set includes the cooking actions GET-KITCHEN, FETCH-AND-PROPORTION, BRING-TO-TEMPERATURE, TRANSFER-CONTENTS, and BEAT. Arrows between variables denote shared arguments between the cooking actions, pointing from output arguments to input arguments.

simulators, and (iv) a standardised evaluation procedure. The benchmark task consists in mapping a recipe expressed in natural language to a set of cooking actions that is precise enough to be executed in the kitchen simulator and thereby produces the desired dish. An example of such a mapping is presented in Figure 3.1. Crucially, our evaluation procedure does not compare the resulting set of cooking actions against a gold standard annotation. Instead, we evaluate how close the dish obtained by executing the cooking actions in the simulated kitchen resembles the gold standard dish. Solving this benchmark task thus requires to integrate reasoning over the recipe text, the state of the simulated kitchen, common-sense knowledge, knowledge of the cooking domain, and the action space of a virtual or robotic chef.

While the core task of this benchmark is similar to semantic parsing, it goes beyond that in two ways. First, in order to facilitate situated reasoning, the provided kitchen simulators can be accessed at any point in the process of mapping the recipe text to cooking actions. The use of kitchen simulators, rather than an actual robotic kitchen, allows to focus on a broader range of instructions and on the high-level logic underlying their execution, rather than on the lower-level robotic control systems needed to implement the task. Second, as the same dish can be prepared in many different ways, we compare the prepared dish against the gold standard dish, instead of comparing the resulting semantic specification directly. Given these reasons and the fact that this task cannot be cast to any other standard NLP task, no baseline results using off-the-shelf techniques are provided.

The benchmark does not take the form of a typical machine learning benchmark, as its goal is not to predict gold standard annotations directly. Instead, we provide the entire corpus of 30 recipes annotated with sets of cooking actions together with the ontology and the kitchen simulators as a testbench, without specific training and test sets. Participants to the benchmark are free to determine how much of this data is used for development, training, validation, setting hyperparameters, designing prompts, or whatever is necessary for their particular approach to tackling the benchmark. Next to the benchmark components that we provide, participants are encouraged to use external data, such as general knowledge graphs (e.g. Wikidata [37]), domain-specific ontologies (e.g. FoodOn [13] or BAALL [21]) or large language models (e.g. GPT-3 [5]). Our benchmark is accessible online at <https://ehai.ai.vub.ac.be/recipe-execution-benchmark/>.

The remainder of this paper is structured as follows. Section 3.2 presents the different components of the benchmark in more detail. Section 3.3 provides an illustrative example of the benchmark task. An overview of related work is presented in Section 11.5. Finally, Section 3.5 provides a concluding discussion.

3.2 The Recipe Execution Benchmark

The recipe execution benchmark consists of four main components: a recipe corpus of 30 recipes (Section 3.2.3), a procedural semantic representation language specifying 38 cooking actions (Section 3.2.4), qualitative and quantitative kitchen simulators (Section 3.2.5), and a standardised evaluation procedure (Section 3.2.6). Before addressing those, we first elaborate on the kitchen state data structure (Section 3.2.1) and the cooking ontology (Section 3.2.2).

3.2.1 Kitchen States

The kitchen state is a central data structure in the recipe execution benchmark. It offers a complete specification of the simulated kitchen, regardless of which simulator is used (cf. Section 3.2.5). The operationalisation of all cooking actions in the procedural semantic representation language (cf. Section 3.2.4) essentially boils down to manipulating the kitchen state in some way. Kitchen states, like all other entities in the simulated kitchen, can be represented as feature structures, i.e. (possibly nested) sets of attribute-value pairs.

The kitchen itself has an ambient temperature of 18°C. The main food preparation area is the counter top. Apart from this, the kitchen also has an oven, a stove, a microwave, a freezer (at -18°C), a fridge (at 5°C), a pantry and a kitchen cabinet. The ingredients are stored in the freezer, the fridge, and the pantry, while the kitchen cabinet is used to store all tools and utensils.

The execution of a recipe always starts from an initial kitchen state, containing all ingredients, utensils, and appliances that are necessary to be able to prepare the recipes provided with the benchmark. The initial kitchen state can be acquired by executing the cooking action GET-KITCHEN (cf. Section 3.2.4). A graphical overview of the initial kitchen state is provided in the technical appendix.

3.2.2 Cooking Ontology

Knowledge about the cooking domain is captured in an ontology that specifies the type system underlying the procedural semantic representation language, the kitchen states, and the kitchen simulators. Specifically, it defines kitchen states, ingredients, tools, modes, and units and relates them in a hierarchical structure. For instance, the ontology is used to indicate (i) that ‘brown sugar’ and ‘white sugar’ are types of ‘sugar’, (ii) that, among others, the freezer, the fridge, and the kitchen cabinet, but also bowls and cookie trays are types of containers, but the latter are transferable containers, while the former are not, and (iii) that certain objects are beatable, boilable, brushable, crackable, cuttable, mashable, meltable, perishable, shakeable, washable, etc. This ontology builds further on the SOMA ontology [3], which aims to capture the physical and the social context of everyday activities and in itself extends the DUL ontology [23]. A full specification of the cooking ontology in YAML format is provided in the technical appendix.

3.2.3 Recipes

The benchmark dataset consists of 30 English recipes sourced from five different websites: AllRecipes.com (<https://www.allrecipes.com>), SimplyRecipes.com (<https://www.simplyrecipes.com>), Food.com (<https://www.food.com>), TheSpruceEats.com (<https://www.thespruceeats.com>), and Cooks.com (<https://www.cooks.com>). The recipes fall in two different categories: 15 recipes for preparing cookies and pastries, and 15 recipes for making salads. All recipes are relatively straightforward for a human to prepare, as they involve a limited number of ingredients, execution steps and required utensils. The recipe texts cover a variety of linguistic phenomena, such as co-references, anaphoric expressions, null arguments, ellipses, hyponyms, meronyms, etc.

The recipes are provided in XML format. For each recipe in the testbench, a set of actions that leads to a perfect end result is provided in the technical appendix. This is for illustrative purposes only, as there may exist other sets of actions that might lead to the same result.

3.2.4 Procedural Semantic Representation Language

For the purpose of this benchmark, we have designed a representation language in terms of procedural semantics [39, 47, 25], i.e. semantic representations that can be executed algorithmically. The procedural semantic representation language defines 38 cooking actions that can be executed in the simulated kitchen environments. Cooking actions are represented as predicates that can be declaratively combined by sharing their variable arguments (see Figure 3.1).

Cooking actions capture atomic operations that an agent should be able to perform in a kitchen setting. The 38 cooking actions that we provide are sufficient to cover at least all recipes in the benchmark. The actions can be grouped in six broad categories: (i) obtaining the kitchen state (GET-KITCHEN), (ii) location altering (e.g. FETCH-AND-PROPORTION and TRANSFER-CONTENTS), (iii) food combination (e.g. MIX, BEAT, SPREAD), (iv) food separation (e.g. SIFT, CRACK, CUT), (v) food manipulation (e.g. MELT, BOIL, WASH) and (vi) tool manipulation (e.g. COVER, LINE, GREASE). These actions are specified at a level of abstraction that allows to focus on the high-level logic of each action, rather than the lower-level robotic control systems needed to implement the action. The implementations are provided by the simulation environment that is used for evaluation (see Section 3.2.5).

Arguments of cooking actions are always typed according to the types defined in the cooking ontology (cf. Section 3.2.2). In general, types of arguments of cooking actions can be divided in five broad categories: (i) kitchen states, (ii) food, (iii) tools, (iv) mode specifiers (e.g. cutting patterns, arrangement patterns, etc.) and (v) quantities. Every cooking action has at least an input kitchen state and an output kitchen state as its arguments. The input kitchen state is always specified after the output kitchen state. Arguments following the input kitchen state are considered input arguments, while arguments preceding the output kitchen state are considered output arguments. The technical appendix provides a detailed technical specification of all cooking actions.

3.2.5 Simulation

The kitchen simulator is used to execute cooking actions. The simulator has two modes through which actions can be performed: qualitative simulation and quantitative simulation. The qualitative simulation operates on the symbolic level, while the quantitative simulation contains a physics engine that is able to model the physical properties of ingredients and actions. Both

simulation modes receive the same initial kitchen state (Section 3.2.1), have the same underlying ontology (Section 3.2.2), and implement the same cooking actions (Section 3.2.4).

The execution of individual cooking actions consists in finding bindings, i.e. computing values, for the output arguments given some values for the input arguments. A value can only be bound to an argument of a cooking action if the type of that value is the same or a subtype of the type specification of the argument (cf. Section 3.2.2). The input kitchen state of the cooking action represents the state of the kitchen before the action has been executed, while the output kitchen state is a copy of the input kitchen state where the effects of the cooking action have taken place. The remaining input and output arguments are the kitchen entities being manipulated or created by the cooking action.

Executing an entire set of cooking actions thus consists in finding the order in which individual cooking actions can be executed depending on the available bindings. The result is a list of bindings from all variable arguments in the provided cooking actions to specific kitchen entities, thereby grounding the actions in the simulation environment.

Some cooking actions implement a form of default reasoning on the level of their arguments. For instance, the last input argument of the cooking action BEAT is a ?BEATING-TOOL. If no value for this argument is provided, the cooking action can still be executed. In that case, a whisk will be fetched from the kitchen cabinet and used for the beating action. However, if the recipe specifies to beat egg whites with a fork, the aim should be to bind an instance of a fork to the ?BEATING-TOOL argument.

Temporal dependency relations between cooking actions are modelled by keeping track of the simulation time. Concretely, each cooking action specifies the time when the output arguments become available in the simulator based on the time of availability of its input arguments and the duration of the cooking action. Using this information, it can be discovered which cooking actions can be executed in sequence or in parallel.

The kitchen simulator is built using the Incremental Recruitment Language (IRL) system [34, 24, 26]. IRL is a formalism for operationalising procedural semantics. It provides the necessary abstractions for defining structured representations of the environment (here, kitchen entities) and primitive operations (here, cooking actions), as well as for executing sets of primitive operations w.r.t. the environment in the manner described above. A key benefit of IRL is that it allows full control over the implementation of each cooking action. For the purpose of our kitchen simulator, it is used as an execution engine that delegates the execution of each cooking action to either the qualitative or quantitative simulation.

The kitchen simulators can be accessed through API calls. Each call should specify the execution mode, i.e. qualitative or quantitative, and the set of cooking actions to be executed. The resulting bindings are represented as a dictionary where the keys are the variables in the cooking actions and the values are objects from the simulation environment encoded in the JSON format.

Qualitative Simulation The aim of the qualitative simulator is to model the kitchen environment on a qualitative level (see e.g. [22, 8]), thereby abstracting away from the fine-grained, numerical details of what for example physics simulators offer. When a cooking action is executed in this symbolic kitchen environment, it causes one or more meaningful changes in the kitchen state. For instance, the cooking action (FETCH-AND-PROPORTION ?PROPORTIONED-BUTTER, ?KS-WITH-BUTTER, ?KITCHEN, ?TARGET-CONTAINER-1, BUTTER, 230, G) from Figure 3.1 finds an empty bowl in the kitchen cabinet and detects butter in the fridge, places both the bowl and the container of butter on the counter top, transfers 230g of that butter to

the empty bowl, and puts the remainder of the butter back into the fridge. This leads to a new kitchen state (?KS-WITH-BUTTER) that is a copy of the previous kitchen state (?KITCHEN) with the following changes: (i) the butter container in the fridge contains 230g less butter, (ii) an empty bowl (?TARGET-CONTAINER-1) has been removed from the kitchen cabinet, and (iii) the counter top contains a bowl with 230g of butter (?PROPORTIONED-BUTTER). An example kitchen state obtained by executing all cooking actions from Figure 3.1 is provided in Figure 3.2. In this figure, only the counter top is visualised and only the large bowl with homogeneous mixture is fully expanded.

Quantitative Simulation In contrast to the qualitative simulation, the quantitative simulation models the physical properties of ingredients and manipulation of objects in more detail. This simulation is built on top of the widely adopted PyBullet robotics simulator package [?]. In turn, PyBullet uses OpenGL and benefits from – but does not require – hardware acceleration via GPUs. A screenshot of the kitchen environment in the qualitative simulation is provided in Figure 3.3.

In executing cooking actions, lower-level details about the execution, such as where exactly to place an object, are decided by the simulation itself. Simulation in PyBullet covers rigid body physics. The simulation environment is three-dimensional, with objects being represented by low-polygon count meshes. Liquids are approximated by collections of particles. Other aspects, such as baking or cutting, are approximated via so-called custom dynamics. These are scripts running in parallel to the simulation that check whether triggering motions occur. For cutting, for instance, a triggering motion is when the blade part of an object that can cut (determined via the ontology) approaches an object that is cuttable (determined via the ontology). As a result, the motion event triggers and the original object is replaced by a collection of separated parts.

Next to all object properties from the qualitative simulation, the kitchen states in the quantitative simulation keeps a full description of the physical state of objects in terms of position, orientation, linear and angular velocity, and any other state variables that are relevant to the custom dynamics implemented via scripting. The quantitative simulation thus complements the qualitative simulation in that it provides information about the influence of general physics on the kitchen state rather than about higher-level qualitative state changes.

3.2.6 Evaluation Procedure

The focus of our evaluation procedure lies on measuring the successful execution of the cooking actions obtained by processing the recipe text. We introduce a simulation-based metric called the dish approximation score (Section 3.2.6). Together with this metric, the quality of the obtained cooking actions can be gauged using a semantic similarity measure (Section 3.2.6) and the kitchen simulator’s execution time (Section 3.2.6). Finally, we provide a graphical evaluation tool that allows participants of the benchmark to easily compute these metrics (Section 3.2.6).

Dish Approximation Score The dish approximation score is inspired by existing metrics used to gauge the performance of compositional directives with non-reversible state changes [?]. With this score, we aim to quantify how similar two dishes are independently from the steps involved in their preparation.

The dish approximation score is computed for all objects in the final kitchen state and the maximum score is returned. This is because the output argument of the final cooking action

does not necessarily correspond to the final dish, as other operations might still be executed after the dish is ready, e.g. putting away utensils or cleaning up the kitchen. However, this approach yields satisfactory results since the final dish will always be present somewhere in the final kitchen state, e.g. one of the kitchen entities on the counter top.

The dish approximation score is computed by comparing the attributes and values of a given kitchen entity and the gold standard dish (see Figure 3.4). The final score is a weighted sum where 2% of the points count towards the presentation (e.g. where the dish is located and how many portions are prepared) and 98% of the points count towards the contents. To compare the dish’s contents, we compare the food products the dish is made of and in which quantities. Each food product may be composed of a combination of other food products. These are first decomposed until the base ingredients are reached. Base ingredients are the food products that cannot be further decomposed in terms of the level of granularity of the benchmark. For instance, in Figure 3.4, the HOMOGENEOUS-MIXTURE-2 in the prepared dish consists of COCOA-POWDER, ALL-PURPOSE-FLOUR, VANILLA-EXTRACT, and HOMOGENEOUS-MIXTURE-1, which itself is composed of WHITE-SUGAR and BUTTER.

The computation of the dish approximation score starts from the base ingredients. Concretely, for every base ingredient in the gold standard dish, we look up the most similar base ingredient in the prepared dish. The score for this base ingredient is the number of attributes and values both ingredients have in common, divided by the total number of attributes and values of the gold standard ingredient. In addition to the properties of the base ingredients, we also check the sequence of intermediate food products they are used in, and how many properties these intermediate food products have in common. Gold standard ingredients that are missing in the prepared dish or prepared ingredients that are in excess are given a similarity score of zero. The similarity scores are again combined by a weighted sum, where 60% counts towards the similarity of the base ingredients and 40% counts towards the similarity of the intermediate food products. All weights involved in computing the dish approximation score were chosen to reflect human intuition after small-scale experiments with the 30 benchmark recipes. Figure 3.4 highlights differences in attribute values that negatively influence the dish approximation score. Note that the entities shown in Figure 3.4 are for illustrative purposes and that the actual entities used in the simulation environments generally have more properties.

Smatch Score A direct comparison of the cooking actions performed by an artificial agent with some example annotation can already give a first indication of how closely the end result will resemble the gold-standard dish. To this end, we provide a modified version of the Smatch score [10] that first converts the cooking actions from the procedural semantic representation language to triples before computing the maximum of F-scores. However, as the Smatch score is based purely on the semantic structure, its value decreases whenever some cooking actions are performed in a different order even though the end result remains the same. As this frequently occurs in recipes, the value of the Smatch score should always be interpreted as an indication together with the other metrics.

Recipe Execution Time The recipe execution time measures the number of timesteps that are needed to execute a set of cooking actions in terms of the kitchen simulator’s internal clock. This metric can be used to gauge cooking efficiency, which could be an indication of insight into the recipe. However, a lower execution time does not necessarily indicate better performance on the benchmark as it can be caused by insufficient understanding of the recipe text, thereby performing less or incorrect cooking actions. Thus, the execution time should always be interpreted in light of the other evaluation metrics.

Evaluation Tool The benchmark comes with an evaluation tool that can be used to obtain the aforementioned metrics. The evaluation tool expects a single input file that specifies the unique identifier of a recipe (e.g. #ALMOND-CRESCENT-COOKIES) followed by the set of all cooking actions that is obtained by processing the entire recipe text. Each cooking action must be specified on a separate line. Through the use of recipe identifiers, the same input file can be used to evaluate multiple recipes. The evaluation tool produces a single file containing the evaluation metrics for each recipe on a separate row. The evaluation tool also allows to visualise the entire simulation process on a web interface.

3.3 Illustrative Example

Figure 3.5 provides an illustrative example of one possible approach for tackling the benchmark task. In this approach, the recipe text is processed line per line, interleaved with the execution of the resulting cooking actions in the qualitative or quantitative kitchen simulator. This way, the approach makes maximum use of the kitchen simulators to facilitate situated reasoning over the recipe. At the start of the recipe, a binding of the initial kitchen state to the variable ?INITIAL-KS is obtained by executing the action GET-KITCHEN. The kitchen state ?INITIAL-KS together with any other sources of information, such as the cooking ontology, the action space of the artificial chef, general knowledge graphs, large language models, etc., can be used to map the instruction “*230 grams of butter, room temperature*” to a set of possible cooking actions. In this example, this results in the cooking actions FETCH-AND-PROPORTION and BRING-TO-TEMPERATURE. Note the use of the variable ?INITIAL-KS in the action FETCH-AND-PROPORTION. This action thus operates over the initial kitchen state. Next, these two cooking actions are executed using either the qualitative or quantitative kitchen simulator. This results in a new kitchen state ?KS-WITH-BUTTER, which includes a bowl with 230 grams of butter at room temperature ?WARM-BUTTER. Now, this new kitchen state, the bowl of butter and any other information can be used in processing the next instruction of the recipe, yielding another set of cooking actions, which is then executed, and so on. All kitchen entities in the final kitchen state ?KS-6 are compared against the gold standard dish and the kitchen entity with the highest dish approximation score is returned.

Given this example approach, we draw the attention to two aspects that are crucial when tackling the recipe execution benchmark. First, there is the challenge of mapping the instructions from the recipe text to the appropriate set of cooking actions. This requires reasoning over the recipe text, the current state of the simulated kitchen, common-sense and domain-specific knowledge and the action space of the virtual or robotic chef. For example, in order to beat the butter and sugar together, these ingredients need to be combined in a single container. From the previous kitchen state, it can be derived that there are bowls of proportioned sugar and butter on the counter top. Hence, two TRANSFER-CONTENTS actions are required to transfer both sugar and butter in a new bowl before the BEAT action can be executed. Second, notice how the processing of individual utterances results in sets of cooking actions that re-use the same variables. These variables are highlighted in red in Figure 3.5. Concretely, the variable ?KS-WITH-WARM-BUTTER in the second network was introduced in the first network, while ?KS-BUTTER-SUGAR, ?WARM-BUTTER, and ?BOWL-WITH-SUGAR are variables in the third network originating from the first and second network. This re-use of variables is essential for carrying out the cooking actions correctly, as, for example, a kitchen state with the right amount of butter and sugar is required for processing and executing the instruction “*Beat the butter and sugar together*”. Indeed, in terms of language processing, the use of “*the butter and sugar*” indicates that these referents were introduced before, while in terms of execution, these entities need to be found in the current kitchen state. In general, the re-use of variables allows

cooking actions to refer back to previous cooking actions or their execution results (as in “*the dough*” in Section 3.1). From these examples, it is clear that the proposed benchmark task goes beyond semantic parsing tasks. In this benchmark, the processing of the recipe text can only yield correct networks of cooking actions when combined with information from previous recipe instructions, their execution in the simulated kitchen, common-sense and domain-specific knowledge, and the action space of the virtual or robotic chef. The main challenge of the benchmark thus lies in the integration of these various components and reasoning over the different sources of knowledge.

An additional, interactive example can be accessed via <https://ehai.ai.vub.ac.be/demos/recipe-understanding/>. The semantic parsing part is here operationalised using the Fluid Construction Grammar framework [36, 44, 4, 5].

3.4 Benchmarks for Recipe Execution

The aim of the recipe execution benchmark proposed in this paper is to assess the ability of an artificial agent to understand and reason about natural language recipes. Closely related to this aim are the tasks of semantic parsing of recipe texts (Section 3.4.1) and the robotic execution of recipes (Section 3.4.2).

3.4.1 Semantic Parsing for Recipes

There exists a wide variety of semantic annotation schemes that aim to address the challenges commonly found in the semantic parsing of recipe texts.

The Carnegie Mellon University Recipe Database (CURD) [33] consists of 260 English recipes annotated with the Minimal Instruction Language for the Kitchen (MILK) annotation scheme. MILK consists of 12 high-level cooking actions expressed as first-order logic predicates. The execution of these operations modifies a symbolic kitchen state that keeps track of ingredients and tools. [33] report only preliminary results on parsing the recipe text into the correct sequence of predicates, without predicting the predicates’ arguments. More recently, LLMs were used for tackling this particular semantic parsing task [11].

The Simplified Ingredient Merging Map in Recipes annotation scheme (SIMMR) uses the MILK annotations of the CURD dataset to generate a dependency tree that related recipe instructions to either previous instructions or ingredients [16]. It thereby offers a more high-level, but also more coarse-grained representation of the structure of recipes compared to MILK.

With the aim of moving away from a domain-specific annotation scheme, the Recipe Instruction Semantics Corpus (RISeC) [17] annotates the CURD recipes with a frame-based annotation scheme, using PropBank frames [20]. Co-reference links across sentences as well as natural language description of implicit references to earlier concepts are added. BERT-based models are used for entity recognition, relation extraction, and zero anaphora identification tasks.

[19] annotated recipes using so-called action graphs. These graphs are composed of predicates that share some of their arguments. The predicates are tied to verbs in the recipe, while the arguments are tied to food items and locations. They annotated a corpus of 2456 recipes. The authors propose an unsupervised machine learning approach using EM algorithms.

Finally, recipes are annotated using flow graphs. Based on initial work by [25], the Recipe Flow Graph (r-FG) Corpus [24] consists of 266 Japanese recipes annotated with predicate-argument structures. These graphs combine syntactic relations (e.g. subj, d-obj, i-obj) with cooking-specific relations (e.g. food part-of, tool complement) to represent the structure of a recipe in

a single graph. Later, an English Recipe Flow Graph Corpus of 300 recipes was released by [40] and semantic parsers for this corpus have recently been presented by [12] and [14]. Multi-modal annotations using recipe flow graphs are also being provided by grounding the nodes of the graph in image data using bounding boxes [27] or image pairs [29].

3.4.2 Systems for Recipe Execution

Systems for recipe execution combine robotic manipulation, computer vision, knowledge representation and reasoning into an integrated approach.

Neural Process Networks [7] use a neural network architecture to integrate cooking actions and entities. They model the understanding of a recipe as finding a sequence of kitchen state changes induced by executing actions on kitchen entities in simulation. This simulation is performed by updating and tracking a set of a priori specified actions and entity embeddings that encode information along six relevant dimensions, namely location, cookedness, temperature, composition, shape and cleanliness.

The MIT BakeBot project [6] operationalises the execution of different recipes by a physical robot. From a dataset of recipes annotated with sequences of states and actions, BakeBot first learns a policy in simulation. To cook a dish, BakeBot then uses this policy to map the recipe instructions to a sequence of states and actions, after which they are executed on a physical robot. However, each action first needs to be translated to multiple, low-level motion operations. The instructions that the robot can execute are rather limited due to the need to implement them in physical reality and some actions require human assistance to be executed.

Moving away from cooking instructions, the CRAM cognitive architecture presented by [2, 1] enables a variety of physical robots to complete everyday manipulation tasks, such as setting the table. The central principle of their architecture is the ability to propagate information both bottom-up, from action and perception modules, and top-down, from high-level planning and meta-level reasoning components to a central declarative reasoning system.

Finally, [15] focus on everyday activity commands in the context of household robotics. They have designed a processing pipeline consisting of components for semantic parsing, language grounding, and simulation, with an overarching ontological framework. However, their pipeline currently only handles single directives in isolation.

3.5 Conclusion

We have presented a novel benchmark that has been designed as a test bed for evaluating whether artificial agents are able to understand how to perform everyday activities in the cooking domain. Understanding how to execute recipes is a highly challenging endeavour due to the underspecified and grounded nature of recipe texts and the fact that recipe execution is a knowledge-intensive and precise activity. The benchmark task that we propose consists in mapping recipes specified in natural language to a set of cooking actions that is concrete and detailed enough to be executed in a simulated kitchen environment, while also producing the desired dish. To this end, we provided (i) a corpus of 30 recipes, (ii) a procedural semantic representation language of 38 cooking actions, (iii) qualitative and quantitative kitchen simulators, and (iv) a standardised evaluation procedure. To overcome the challenges inherent to recipe execution, the mapping of recipe texts to cooking actions requires to integrate reasoning over the recipe text, the state of the simulated kitchen, common-sense knowledge, knowledge of the cooking domain, and the action space of a virtual or robotic chef. This benchmark distinguishes itself from other semantic parsing and recipe execution benchmarks in that (i)

kitchen simulators can be used at any time to facilitate situated reasoning and (ii) evaluation is not in terms of a gold-standard annotation, but in terms of the quality of the prepared dish. The benchmark thereby addresses the growing interest in human-centric systems that combine natural language processing and situated reasoning to perform everyday activities [32, 36].

counter-top-256-1		
name: nil		
persistent-id: counter-top-256		
simulation-data: nil		
is-concept: nil		
arrangement: side-to-side-538-1		
side-to-side		
whisk-1550-1	large-bowl-530-1	medium-bowl-2980-1
whisk	name: nil	medium-bowl
	persistent-id: large-bowl-530	
	simulation-data: nil	
	used: t	
	is-concept: nil	
	arrangement: nil	
	homogeneous-mixture-4042-1	
	name: nil	
	persistent-id: homogeneous-mixture-4042	
	simulation-data: nil	
	shaken: nil	
	temperature: nil	
	spread-with: nil	
	sprinkled-with: nil	
	is-liquid: nil	
	boiled: nil	
	boiled-with: nil	
	spread: nil	
	dipped-in: nil	
	current-shape: nil	
	flattened: nil	
	baked: nil	
	sifted: nil	
	mixed: nil	
	melted: nil	
	mashed: nil	
	is-cut: uncut-11504-1	
	uncut	
	beaten: t	
	keep-frozen: nil	
	keep-refrigerated: nil	
	is-concept: nil	
	amount: amount-34632-1	
	name: nil	
	persistent-id: amount-34632	
	simulation-data: nil	
	quantity: quantity-52605-1	
	name: nil	
	persistent-id: quantity-52605	
	simulation-data: nil	
	value: 350	
	quantity	
	unit: g-4461-1	
	g	
	amount	
	white-sugar-1614-1	butter-1648-1
	white-sugar	butter
	homogeneous-mixture	
	covered-with: nil	
	large-bowl	
	medium-bowl-2978-1	
	medium-bowl	
	counter-top	

Figure 3.2: Example representation offered by the qualitative kitchen simulator after executing the cooking actions specified in Figure 3.1.

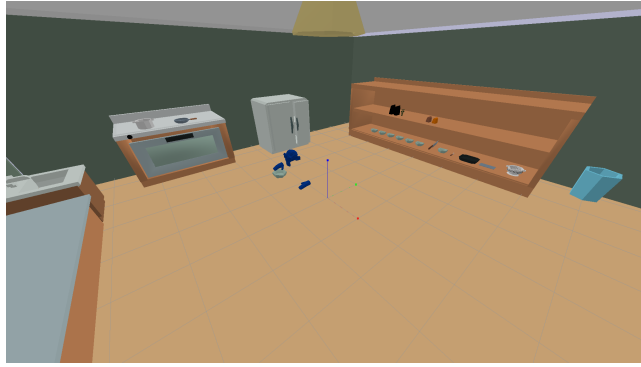


Figure 3.3: Screenshot of the qualitative kitchen simulation environment.

Gold Standard Dish	Prepared Dish
baking-tray <i>located at counter-top</i> - used: true - lined-with: baking-paper - arrangement: side-to-side 25 portions homogeneous-mixture-2 - temperature: 175 °C current-shape: crescent-shape - baked: true mixing-type: mixed - amount: 25 g all-purpose-flour - sifted: false - amount: 10 g vanilla-extract - amount: 1 g homogeneous-mixture-1 - temperature: unspecified - current-shape: unspecified - baked: false - mixing-type: mixed - amount: 14 g white-sugar - temperature: 18 °C - amount: 4 g butter temperature: 18 °C - amount: 10 g	cookie-sheet <i>located at counter-top</i> - used: true - lined-with: baking-paper - arrangement: side-to-side 20 portions homogeneous-mixture-2 - temperature: 175 °C current-shape: ball-shape - baked: true mixing-type: beaten - amount: 36.25 g cocoa-powder - amount: 5 g all-purpose-flour - sifted: false - amount: 12.5 g vanilla-extract - amount: 1.25 g homogeneous-mixture-1 - temperature: unspecified - current-shape: unspecified - baked: false - mixing-type: mixed - amount: 17 g white-sugar - temperature: 18 °C - amount: 5 g butter temperature: 5 °C - amount: 12.5 g

Figure 3.4: Illustrative example of what differences in attribute values negatively impact the dish approximation score (highlighted in red).

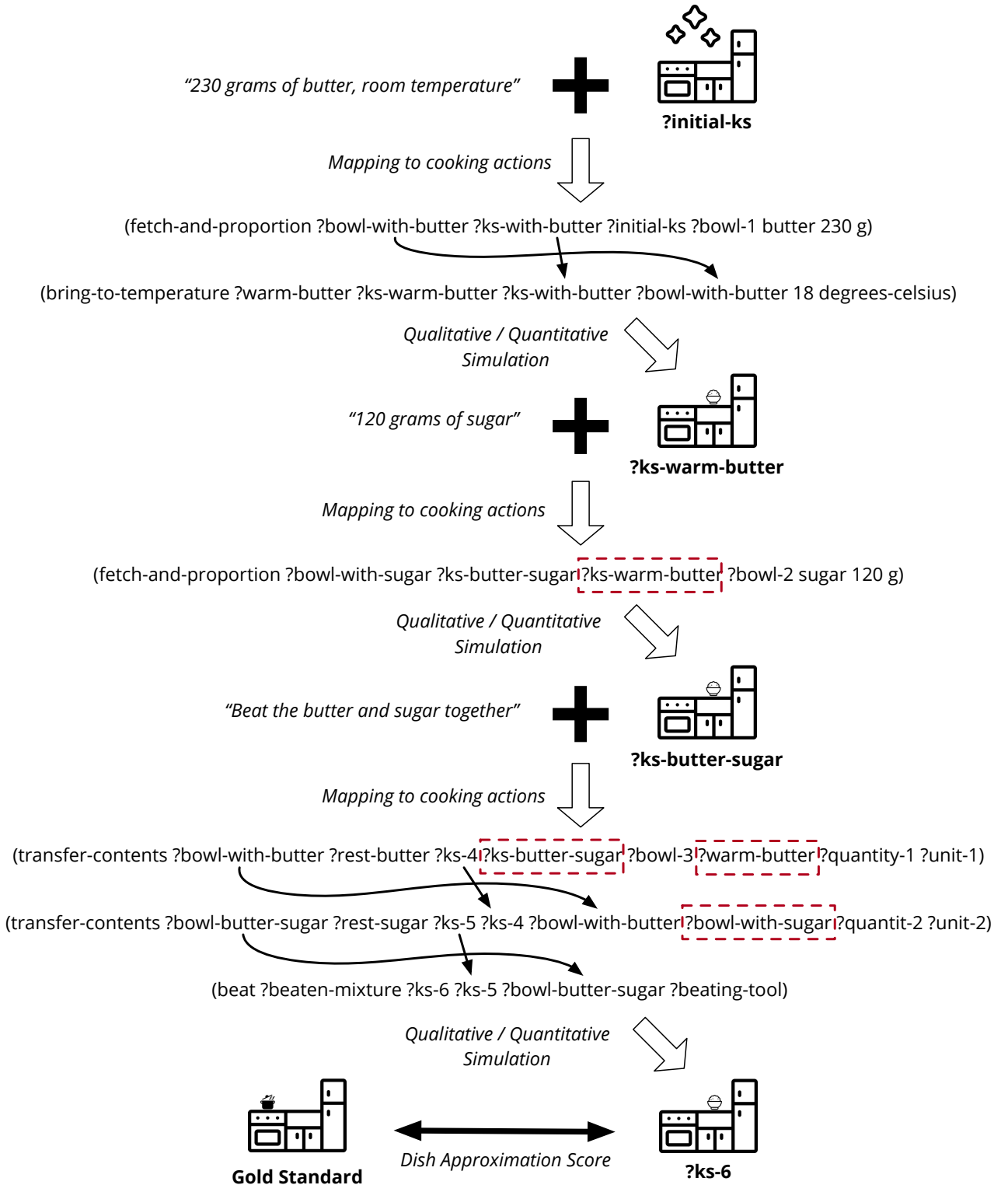


Figure 3.5: Illustrative example of one possible approach for tackling the benchmark task.

Chapter 3: References

- [BA1] Michael Beetz, Gayane Kazhoyan, and David Vernon. The CRAM cognitive architecture for robot manipulation in everyday activities. *arXiv preprint arXiv:2304.14119*, 2023.
- [BA2] Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner, Dejan Pangercic, Thomas Rühr, and Moritz Tenorth. Robotic roommates making pancakes. In *Proceedings of the 11th IEEE-RAS International Conference on Humanoid Robots*, pages 529–536, New York, NY, USA, 2011. IEEE.
- [BA3] Daniel Beßler, Robert Porzel, Mihai Pomarlan, Abhijit Vyas, Sebastian Höffner, Michael Beetz, Rainer Malaka, and John Bateman. Foundations of the socio-physical model of activities (soma) for autonomous robotic agents. In *Formal Ontology in Information Systems*, pages 159–174, Amsterdam, Netherlands, 2021. IOS Press.
- [BA4] Katrien Beuls and Paul Van Eecke. Fluid Construction Grammar: State of the art and future outlook. In *Proceedings of the First International Workshop on Construction Grammars and NLP (CxGs+NLP, GURT/SyntaxFest 2023)*, pages 41–50, Washington, D.C., USA, 2023. Association for Computational Linguistics.
- [BA5] Katrien Beuls and Paul Van Eecke. Construction grammar and artificial intelligence. In Mirjam Fried and Kiki Nikiforidou, editors, *The Cambridge Handbook of Construction Grammar*. Cambridge University Press, Cambridge, United Kingdom, 2024. Forthcoming.
- [BA6] Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. Interpreting and executing recipes with a cooking robot. In Jaydev P. Desai, Gregory Dudek, Oussama Khatib, and Vijay Kumar, editors, *Experimental Robotics: The 13th International Symposium on Experimental Robotics*, pages 481–495. Springer-Verlag, Heidelberg, Germany, 2013.
- [BA7] Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. Simulating action dynamics with neural process networks. In *6th International Conference on Learning Representations (ICLR 2018)*, page 10, 2018.
- [BA8] Ivan Bratko. *Prolog Programming for Artificial Intelligence (Fourth Ed.)*. Pearson Education, Harlow, United Kingdom, 2012.
- [BA9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, Red Hook, NY, USA, 2020. Curran Associates, Inc.
- [BA10] Shu Cai and Kevin Knight. Smatch: an evaluation metric for semantic feature structures. In Hinrich Schütze, Pascale Fung, and Massimo Poesio, editors, *Proceedings of*

the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 748–752. Association for Computational Linguistics, 2013.

- [BA11] Vanya Cohen and Raymond Mooney. Using planning to improve semantic parsing of instructional texts. In Bhavana Dalvi Mishra, Greg Durrett, Peter Jansen, Danilo Neves Ribeiro, and Jason Wei, editors, *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pages 47–58. Association for Computational Linguistics, 2023.
- [BA12] Lucia Donatelli, Theresa Schmidt, Debanjali Biswas, Arne Köhn, Fangzhou Zhai, and Alexander Koller. Aligning actions across recipe graphs. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6930–6942. Association for Computational Linguistics, 2021.
- [BA13] Damion M. Dooley, Emma J. Griffiths, Gurinder S. Gosal, Pier L. Buttigieg, Robert Hoehndorf, Matthew C. Lange, Lynn M. Schriml, Fiona S.L. Brinkman, and William W. L. Hsiao. FoodOn: a harmonized food ontology to increase global food traceability, quality control and data integration. *npj Science of Food*, 2(1):23, 2018.
- [BA14] Yi Fan and Anthony Hunter. Understanding the cooking process with english recipe text. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4244–4264. Association for Computational Linguistics, 2023.
- [BA15] Sebastian Höffner, Robert Porzel, Maria M. Hedblom, Mihai Pomarlan, Vanja Sophie Cangalovic, Johannes Pfau, John A. Bateman, and Rainer Malaka. Deep understanding of everyday activity commands for household robots. *Semantic Web*, 13(5):895–909, 2022.
- [BA16] Jermsak Jermsurawong and Nizar Habash. Predicting the structure of cooking recipes. In Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 781–786. Association for Computational Linguistics, 2015.
- [BA17] Yiwei Jiang, Klim Zaporozhets, Johannes Deleu, Thomas Demeester, and Chris Develder. Recipe instruction semantics corpus (RISeC): Resolving semantic structure and zero anaphora in recipes. In Kam-Fai Wong, Kevin Knight, and Hua Wu, editors, *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 821–826. Association for Computational Linguistics, 2020.
- [BA18] Philip N. Johnson-Laird. Procedural semantics. *Cognition*, 5(3):189–214, 1977.
- [BA19] Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. Mise en place: Unsupervised interpretation of instructional recipes. In Lluís Màrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 982–992. Association for Computational Linguistics, 2015.
- [BA20] Paul R. Kingsbury and Martha Palmer. From treebank to propbank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, pages 1989–1993, Paris, France, 2002. European Language Resources Association (ELRA).
- [BA21] Bernd Krieg-Brückner, Serge Autexier, and Mihai Pomarlan. The BAALL ontology-configuration of service robots, food, and diet. In Michael Grüninger, editor, *FOIS*

- [BA22] Benjamin Kuipers. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. MIT Press, Cambridge, MA, USA, 1994.
- [BA23] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. Wonderweb deliverable d18-ontology library (final). Technical report, National Research Council-Institute of Cognitive Science and Technology, 2003. IST Project 2001-33052 WonderWeb: Ontology Infrastructure for the Semantic Web.
- [BA24] Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. Flow graph corpus from recipe texts. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC)*, pages 2370–2377, Paris, France, 2014. European Language Resources Association (ELRA).
- [BA25] Shinsuke Mori, Tetsuro Sasada, Yoko Yamakata, and Koichiro Yoshino. A machine learning approach to recipe text processing. In *Proceedings of the 1st Workshop on Cooking with Computers*, pages 1–6, Paris, France, 2012. Centre national de la recherche scientifique (CNRS).
- [BA26] Jens Nevens, Paul Van Eecke, and Katrien Beuls. A practical guide to studying emergent communication through grounded language games. In *AISB Language Learning for Artificial Agents Symposium*, pages 1–8, 2019.
- [BA27] Taichi Nishimura, Suzushi Tomori, Hayato Hashimoto, Atsushi Hashimoto, Yoko Yamakata, Jun Harashima, Yoshitaka Ushiku, and Shinsuke Mori. Visual grounding annotation of recipe flow graph. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 12th International Conference on Language Resources and Evaluation*, pages 4275–4284, Paris, France, 2020. European Language Resources Association (ELRA).
- [BA28] Josef Ruppenhofer and Laura A Michaelis. A constructional account of genre-based argument omissions. *Constructions and frames*, 2(2):158–184, 2010.
- [BA29] Keisuke Shirai, Atsushi Hashimoto, Taichi Nishimura, Hirotaka Kameko, Shuhei Kurita, Yoshitaka Ushiku, and Shinsuke Mori. Visual recipe flow: A dataset for learning visual state changes of objects with recipe flows. In Nicoletta Calzolari, Chu-Ren Huang, Hansaem Kim, James Pustejovsky, Leo Wanner, Key-Sun Choi, Pum-Mo Ryu, Hsin-Hsi Chen, Lucia Donatelli, Heng Ji, Sadao Kurohashi, Patrizia Paggio, Nianwen Xue, Seokhwan Kim, Younggyun Hahm, Zhong He, Tony Kyungil Lee, Enrico Santus, Francis Bond, and Seung-Hoon Na, editors, *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3570–3577, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics.
- [BA30] Michael Spranger, Simon Pauw, Martin Loetzsch, and Luc Steels. Open-ended procedural semantics. In Luc Steels and Manfred Hild, editors, *Language Grounding in Robots*, pages 153–172. Springer, New York, NY, USA, 2012.
- [BA31] Luc Steels. Constructivist development of grounded construction grammar. In *Pro-*

- ceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 9–16, 2004.
- [BA32] Luc Steels. Conceptual foundations for human-centric AI. In Mohamed Chetouani, Virginia Dignum, Paul Lukowicz, and Carles Sierra, editors, *Human-Centered Artificial Intelligence (ACAI 2021)*, pages 8–35, Cham, Switzerland, 2023. Springer.
- [BA33] Dan Tasse and Noah A. Smith. SOUR CREAM: Toward semantic processing of recipes. Technical Report CMU-LTI-08-005, Carnegie Mellon University, Pittsburgh, PA, USA, 2008.
- [BA34] Wouter Van den Broeck. Constraint based compositional semantics. In *The Evolution of Language: Proceedings of the 7th International Conference (EVOLANG7)*, pages 338–345. World Scientific, 2008.
- [BA35] Remi van Trijp, Katrien Beuls, and Paul Van Eecke. The FCG Editor. *PLOS ONE*, 17(6):e0269708, 2022.
- [BA36] Lara Verheyen, Jérôme Botoko Ekila, Jens Nevens, Paul Van Eecke, and Katrien Beuls. Neuro-symbolic procedural semantics for reasoning-intensive visual dialogue tasks. In Kobi Gal, Ann Nowé, Grzegorz J. Nalepa, Roy Fairstein, and Roxana Rădulescu, editors, *Proceedings of the 26th European Conference on Artificial Intelligence (ECAI 2023)*, pages 2419–2426, Amsterdam, Netherlands, 2023. IOS Press.
- [BA37] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.
- [BA38] Terry Winograd. Understanding natural language. *Cognitive psychology*, 3(1):1–191, 1972.
- [BA39] William A. Woods. Procedural semantics for a question-answering machine. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, pages 457–471, New York, NY, USA, 1968.
- [BA40] Yoko Yamakata, Shinsuke Mori, and John A. Carroll. English recipe flow graph corpus. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the 12th International Conference on Language Resources and Evaluation*, pages 5187–5194, Paris, France, 2020. European Language Resources Association (ELRA).

Chapter 4

The Proof is in the Almond Cookies: Handling the cooking benchmark using symbolic tools

Remi van Trijp[†], Katrien Beuls^{△,◇}, Paul Van Eecke^{*}

[†]Sony Computer Science Laboratories, Paris Lab

[△]Faculté d’Informatique, Université de Namur

^{*}Artificial Intelligence Laboratory, Vrije Universiteit Brussel

Abstract

This paper presents a case study on how to process cooking recipes (and more generally, how-to instructions) in a way that makes it possible for a robot or artificial cooking assistant to support human chefs in the kitchen. Such AI assistants would be of great benefit to society, as they can help to sustain the autonomy of aging adults or people with a physical impairment, or they may reduce the stress in a professional kitchen. We propose a novel approach to computational recipe understanding that mimics the human sense-making process, which is narrative-based. Using an English recipe for almond crescent cookies as illustration, we show how recipes can be modelled as rich narrative structures by integrating various knowledge sources such as language processing, ontologies, and mental simulation. We show how such narrative structures can be used for (a) dealing with the challenges of recipe language, such as zero anaphora, (b) optimizing a robot’s planning process, (c) measuring how well an AI system understands its current tasks, and (d) allowing recipe annotations to become language-independent.

4.1 Introduction

This paper explores what kind of grounded language processing model is needed for enabling robots or computational cooking assistants to support human chefs in the kitchen. Such human-centric AI assistants would be of great benefit for society because they could sustain the autonomy of aging adults or people with a physical impairment, or they could reduce the pressure on professional chefs who have to work in high-stress situations. We propose a novel approach to computational recipe understanding that mimics the narrative-based sense-making process of humans [9], which may lead to more intuitive and meaningful human-robot interactions.



Almond Crescent Cookies

PREP TIME 15 mins	COOK TIME 20 mins	TOTAL TIME 35 mins
-----------------------------	-----------------------------	------------------------------

SERVINGS
30 servings

Ingredients

- 226 grams butter, room temperature
- 116 grams sugar
- 4 grams vanilla extract
- 4 grams almond extract
- 340 grams flour
- 112 grams almond flour
- 29 grams powdered sugar

Instructions

- Beat the butter and the sugar together until light and fluffy.
- Add the vanilla and almond extracts and mix.
- Add the flour and the almond flour.
- Mix thoroughly.
- Take generous tablespoons of the dough and roll it into a small ball, about an inch in diameter, and then shape it into a crescent shape.
- Place onto a parchment paper lined baking sheet.
- Bake at 175°C for 15-20 minutes.
- Dust with powdered sugar.

Figure 4.1: This Figure shows an English recipe for almond crescent cookies, adapted from https://www.simplyrecipes.com/recipes/almond_crescent_cookies/.

We first discuss the main challenges of recipe understanding and related work before introducing **narrative-based understanding** (section 4.2, also see [58]). We then illustrate the approach through a concrete case study on an English recipe for almond crescent cookies, shown in Figure 4.1. Finally, we evaluate the benefits and the scalability of the approach, and provide more information about resources made available to the community for researchers who wish to experiment with narrative-based understanding.

Challenges

Recipe understanding is a challenge for robotics because kitchens are rich and dynamically changing environments [8]. From a linguistic perspective, recipes come with their own genre-specific syntax and semantics (see a.o. [14, 20, 12]) that challenge traditional NLP solutions, of which we summarize the most important ones here:

- *How-to instructions*: Recipes use procedural language, such as imperative commands, which leads to reduced performance of off-the-shelf parsers [53].
- *Zero anaphora*: Recipes are abundant with zero anaphora (e.g. no direct object in the phrase “mix thoroughly”) because cooking takes place in an actual kitchen that provides the necessary context for filling in the blanks.
- *Dynamic Environment*: Kitchens are dynamic environments in which entities are changed into “resultant objects” – often without explicit mention of that happening. For instance, the almond cookie recipe (Figure 4.1) introduces the phrase “the dough” for the first time in its fifth instruction without making explicit that it is the resultant object of mixing together various ingredients such as butter, sugar and flour.
- *Complex Semantics*: Recipes require careful management of time, measurement and ordering. Instructions can be explicit (such as “340 grams flour” or “for 15-20 minutes”), but recipes also often use vague measurements (“generous tablespoons”) and evaluative phrases (“until light and fluffy”) that require a tight integration of language processing

and sensorimotor perception.

4.1.1 Related Work

Computational recipe understanding and other tasks in *Digital Gastronomy* [65] have always enjoyed academic interest (see e.g. the Computer Cooking Contests [36]), but especially in the past few years there has been a surge of attention for the broader field of **food computing** [24, 35]. This surge is driven on the one hand by the explosion of large-scale online data such as recipes and cooking videos; and on the other hand by the breakthroughs in deep learning for handling such large data [32].

Most research therefore focuses on aggregating and cleaning up the data; and on the creation of datasets, benchmarks, representations and classification systems for food-related information [42, 30, 64, 34, 39, 28, 56]. This information is then used for various tasks such as recipe generation [27, 60], recipe recommendation [26, 55], question-answering systems [33, 29], and so on. Ultimately, such systems aim to provide an appropriate response to a particular input, such as proposing relevant recipes based on the user’s preferences.

Even though such work is relevant to the present study, their goals only require a shallow understanding of recipes, while our objective is to parse recipes in such a way that a robot can successfully execute it (or more generally speaking, that the robot can successfully execute instructions). This objective requires adequate systems for **grounded** [23] **natural language understanding** (NLU; [1, 53]).

Despite a longstanding research history going back to the 1970s [62, 25], a recent benchmark study has shown that grounded language understanding is still a largely unsolved problem [41]. That is not to say that no progress has been made: thanks to more sophisticated language technologies and the increasing availability of online data, the field has moved away from limited sets of natural language instructions, and has instead set its ambition on mapping open-ended instructions from the web onto everyday manipulation tasks [54].

In the cooking domain, several prototypes and experiments have been reported [51, 4, 8, 5]. These studies are usually performed from the perspective of robotics, and mainly examine how existing NLP techniques can be repurposed for the generation of executable robot plans [54]. Language processing therefore typically involves translating instructions onto syntactic parse trees from which semantics can be inferred; or more recently, applying neural network models for directly mapping sentences onto formal semantic representations [53].

4.2 Narrative-Based Understanding

We propose to treat recipes as a form of narrative, taking inspiration from discourse-analysis studies in linguistics [14] and recent work on the value of narratives for human-centered AI [52, 6, 49, 58]. Narratologists divide a narrative into three interconnected layers [3], illustrated in Figure 4.2:

1. The *fabula* (or *story*) is a collection of facts, events and actions;
2. The *plot* (or *syuzhet*) is a structure that arranges the relevant items of the fabula in a causal network of events that lead to a conclusion (called *narrative closure* [13]);
3. The *narration* (or *discourse*) is how the narrative is presented.

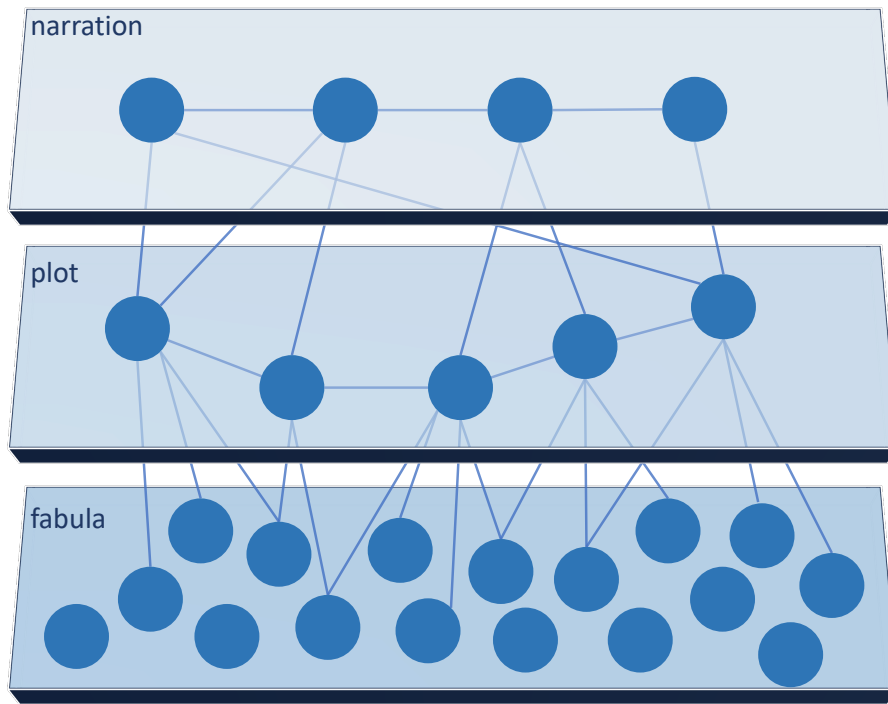


Figure 4.2: A narrative is a three-layered structure consisting of a fabula, plot, and narration. Narrative-based understanding involves constructing the plot using the narration and the fabula, thereby integrating language processing, memory, mental simulation, perception, and so on.

4.2.1 Recipes as Narratives

Let us apply these three layers to recipes-as-narratives. There are two observable layers: the *fabula* and the *narration*. The *fabula* is so vast (i.e. most of its content is irrelevant such as ingredients that won't be used) that a cooking agent can only maintain a partial model, which it obtains through sensorimotor processing and retrieving facts from memory (e.g. which drawer contains the cutlery). While the fabula can be considered as the background against which the narrative should be situated, the *narration* concerns how the narrative is presented, which can be a written recipe, a cooking video, a dialogue, and so on. This layer is typically analyzed using (multimodal) language processing techniques.

At the heart of narrative-based understanding is the *plot*, which is invisible to the cooking agent and which therefore has to be constructed. The plot is a rich content model in which the relevant elements of the fabula are arranged in a causal network of events. By integrating the diverse and often fragmented and ambiguous input from various knowledge sources (such as language processing, vision and pattern recognition, mental simulation, action monitoring, ontologies, knowledge graphs, and so on), the plot provides a coherent and structured path towards the goal of the narrative (in our case study: delicious almond crescent cookies).

In the case of cooking, our main guideline for constructing the causal network of events is the narration. In the simplest case, the narration follows the same order as the plot, but even for recipes there exist many variations [14]. The recipe for almond cookies, for instance, starts with a flash-forward by stating that there will be 30 servings. Other recipes may rely on base recipes (subplots) or include alternative ways to prepare a dish.

Important to note is that the construction of the plot is not the main goal of narrative-based understanding: it rather serves to find *narrative closure* [13], which is the state in which the plot arrives at a satisfactory conclusion. In the case of recipe understanding, narrative closure

is obviously achieved if the desired food is ready to be served.

4.2.2 Language as a Form of Action

Just like narratives involve the active construction of a plot in order to make sense of reality [9], functional theories of linguistics have considered language to be a form of action ever since the influential works of philosophers such as Wittgenstein [63], Austin [2] and Searle [40]. The instructions found in recipes are textbook examples of such *speech acts*: linguistic expressions that invite the addressee to perform a (mental) action.

In robotics and grounded language understanding, those actions take the form of plans that can be simulated or executed by a robot. Traditional approaches typically involve a pipeline going from linguistic expressions to truth-conditional semantic representations [15, 53], which are then mapped onto an executable robot plan. For instance, the phrase “take the dough” can be associated with the logical form $\exists x : \{DOUGH(x) \wedge TAKEN(x)\}$ (“there exists an x that is dough and that is taken”), which using temporal logics [31] can be specified as becoming True once the robot executes the correct operation and takes the dough. This formal semantic specification is then used for generating an executable robot plan [4, 8, 51].

In our approach, we dispense with an intermediary truth-conditional representation and propose that the meaning of a sentence (or indeed, the meaning of the recipe as a whole) **IS** an executable robot plan. For example, the phrase “take generous tablespoons of the dough and roll it into a small ball” in the almond crescent cookie recipe directly maps onto an actual operation in which the cooking agent uses a tablespoon as a tool of measurement for making several spheres made of dough.

4.2.3 Personal Dynamic Memory

Narratives are *personal* as they are based on past experiences, individual beliefs and values, and on which perspective is taken [48, 58]. For instance, if at the world cup football a small nation eliminates one of the tournament’s favourites, their supporters may praise their team’s courage and efficient counter tactics, while the losing side might condemn them for playing a defensive “anti-football” game.

Narratives are therefore not constructed out of the blue, but are integrated into a Personal Dynamic Memory (PDM [48]). A Personal Dynamic Memory consists of persistent knowledge (e.g. an agent’s linguistic inventory, its ontology, and so on) and of past experiences and past narratives. The more cooking experience an agent has acquired throughout its lifetime, the easier it will be able to construct a recipe’s narrative.

4.3 Almond Crescent Cookies

There is an English proverb that goes *the proof is in the pudding*, which comes from the older saying *the proof of the pudding is in the eating*. This expression used to mean quite literally that you have to try out food to know how tasty it is, and nowadays it can be used to say that you can only know the value or quality of something through direct experience or by obtaining concrete results. The same goes for evaluating the value of our narrative-based approach to recipe understanding.

While the previous section offered a conceptual, implementation-independent overview of narrative-based understanding, we will now proceed with a specific operationalization through a concrete case study on an English recipe for almond crescent cookies. This section will make

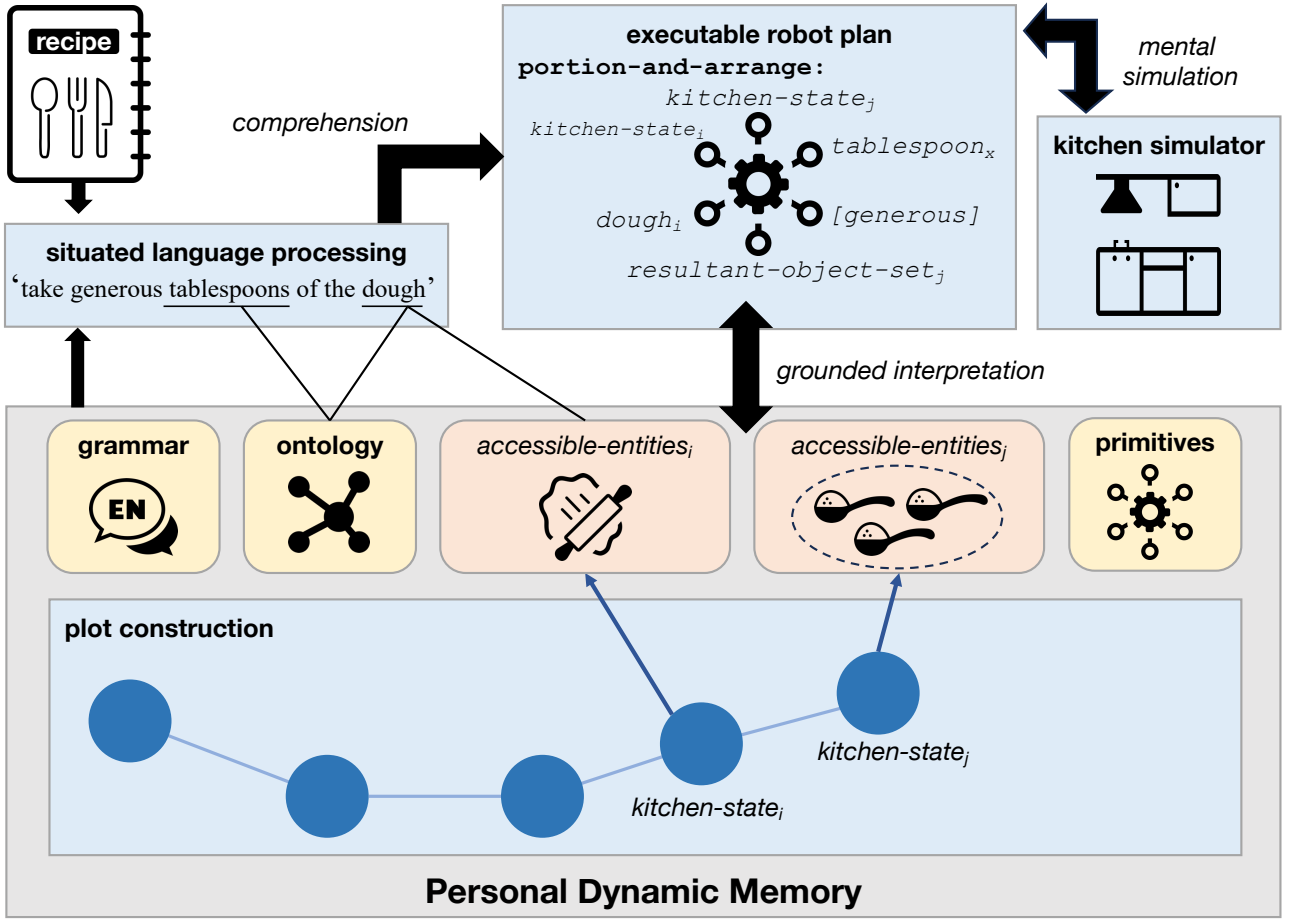


Figure 4.3: This Figure illustrates a single cycle in the construction of the recipe’s plot. On the top left: while parsing, the language processor has access to the cooking agent’s grammar, ontology, and the entities that are currently under its attention (*accessible-entities_i*). Comprehension results in a partial executable robot plan (here the operation *portion-and-arrange*). Through interaction with a kitchen simulator and the agent’s personal dynamic memory, a complete plan is generated and executed, leading to a new plot beat (*kitchen-state_j*), which includes new accessible entities (tablespoons of dough). The cycle can then repeat itself with the next instruction until the recipe is finished.

heavy use of Figure 4.3, which illustrates one cycle of the back-and-forth between language processing, semantic interpretation, mental simulation, and personal dynamic memories. Interested readers can find more technical details at the web demonstration of our case study at <https://see-anonymous-attachment/recipe-understanding/>, and the Recipe Execution Benchmark which we developed for evaluating our approach (see section 4.4).

4.3.1 Situated Language Processing

In order to handle the genre-specific challenges of recipes, we have chosen a Construction Grammar approach [17, 21, 19], which we implemented in the open-source formalism Fluid Construction Grammar [47, 59]. The motivation for this approach is threefold. First, Construction Grammar is a linguistic theory in which *all* linguistic information is represented as mappings between form and meaning (called “constructions”), which makes it convenient to represent both the idiosyncrasies of recipe language as well as its more abstract syntactic structures in a uniform way. Secondly, semantics can but needn’t be directly coupled to syntactic structures, which makes it possible to parse sentences directly into language-independent exe-

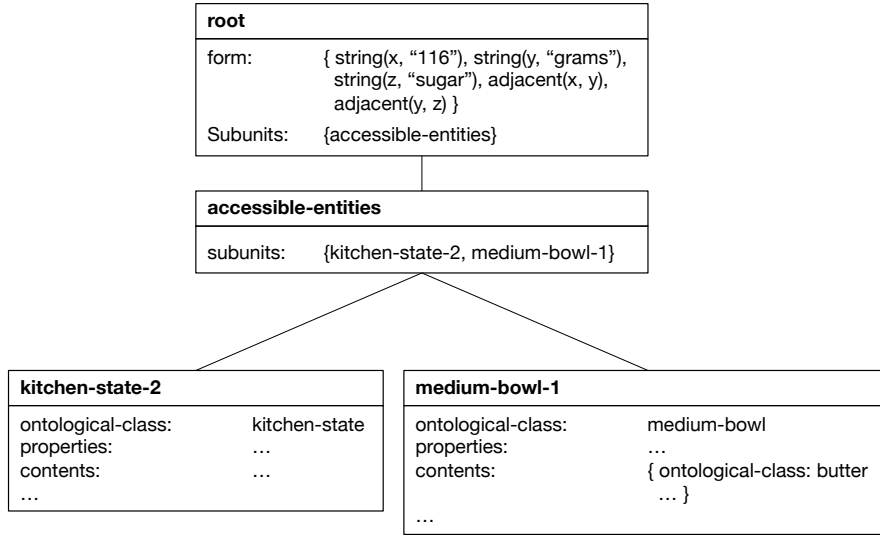


Figure 4.4: A *transient structure* contains information about both the input sentence and the entities that are accessible from discourse context.

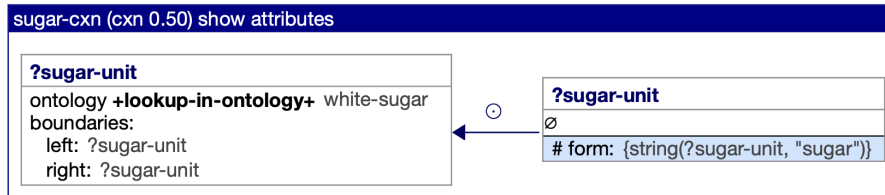
cutable robot plans. Finally, the functional scope of constructions is not limited to the sentence level, which means that constructions can represent discourse-level information as well [18].

The latter feature of Construction Grammar is of great importance for recipes. As discussed in section 4.1, recipes are abundant with zero anaphora, which are used by recipe authors as a strategy for cohesion building since these zero anaphora refer to entities that are highly salient in the current discourse context [12]. For instance, in the almond cookies recipe, the direct object is omitted in phrases such as “mix thoroughly” and “place onto a parchment paper lined baking sheet”.

Our solution is to include non-linguistic information in language processing, which is supplied by the Personal Dynamic Memory as shown in Figure 4.3. The PDM is where the recipe’s plot is constructed: at each node, the PDM tracks which entities are currently under the attention of the cooking agent (called *accessible entities*). Accessible entities are like characters that were introduced in prior scenes and that are still present in the current scene.

More concretely, linguistic processing makes use of a kind of blackboard that contains all of the information about the accessible entities and the input phrase or sentence. This blackboard is called *transient structure* because it changes over time as different constructions access and expand its information. Figure 4.4 illustrates such a transient structure at the beginning of a parsing task for the phrase “116 grams sugar”. This transient structure consists of four units (which are simply lists of feature-value pairs). The unit called *root* (on top) contains unhandled information from the input sentence, such as which strings (or tokens) were observed and which strings are adjacent to each other. At this point in the recipe, the cooking agent has already fetched a medium bowl of butter, so there are two accessible entities: the current kitchen-state, and the bowl of butter.

In Fluid Construction Grammar (FCG), constructions are formalized as schemas that consist of a conditional pole (right-hand side) and a contributing pole (left-hand side). A construction is allowed to add the information of its contributing pole to the transient structure if the feature-value pairs from its conditional pole can be matched against the information already present in the transient structure [50]. Here is an example of a lexical construction for the word “sugar”:



The application of a construction leads to a new and expanded transient structure, which may in turn trigger the application of other constructions. Parsing is thus operationalized as a search problem for finding the best set of constructions for extracting the meaning of an input sentence [57].

Constructions represent most information as declarative features, but they can also use procedural attachment [45, 10]. For instance, in the sugar-construction, the value of the feature *ontology* is not directly specified: instead, a procedure **+lookup-in-ontology+** is embedded, which is able to fetch all of the features that are associated with the concept [white-sugar]. Procedural attachment is necessary for dealing with uncertainty. Some examples are:

1. Human-centric AI systems must be open-ended learners, hence the ontology may change with every novel experience.
2. Higher-level constructions may involve generalizations, e.g. over phrases with ingredients such as “two tablespoons of sugar” or “120 grams of flour”. Procedural attachment allows an on-the-fly check whether sugar or flour both qualify as ingredients.
3. Constructions that handle quantities must be able to parse such units on the fly. For instance, a construction responsible for portions should recognize both the tokens “114” as well as “two”, so a procedure for checking whether a token can be parsed as a number helps to make such generalizations possible.

For the present case study, we hand-coded a small grammar of 56 constructions that can be inspected in our web demo. Hand-coding is a necessary first step to identify what kind of grammatical structures are necessary for mapping recipes onto executable robot plans and for evaluating the feasibility of a constructional approach (we will address the question of learning in section 4.5). Our grammar includes constructions for lemmatization, lexical constructions, idiomatic and semi-schematic constructions (e.g. the “until light and fluffy”- and “place-X-onto-y”-constructions), and abstract constructions such as the English Resultative [7, 22] for analyzing phrases such as “beat the butter and sugar together until light and fluffy”.

4.3.2 Meaning and Mental Simulation

The goal of language processing is not to derive the most accurate syntactic analysis of a sentence: syntactic structures are only built insofar as they help to get to the meaning as efficiently as possible. “The” meaning is a bit misleading because language is an inferential coding system [43] so not all the meaning is in the message. For instance, a phrase such as “put two eggs in a bowl” does not specify which eggs and bowl to take, or that you have to crack the eggs open and get rid of the shells.

Parsing therefore only leads to a partial robot plan that the cooking agent needs to complete and expand upon. For instance, the phrase “116 grams sugar” maps onto an operation called *fetch-and-proportion*, illustrated in Figure 4.5. Some of the operation’s arguments are already provided by the recipe instructions such as which food product to fetch (*sugar*), shown as cyan circles. However, there are several open slots such as the resultant object, shown as red

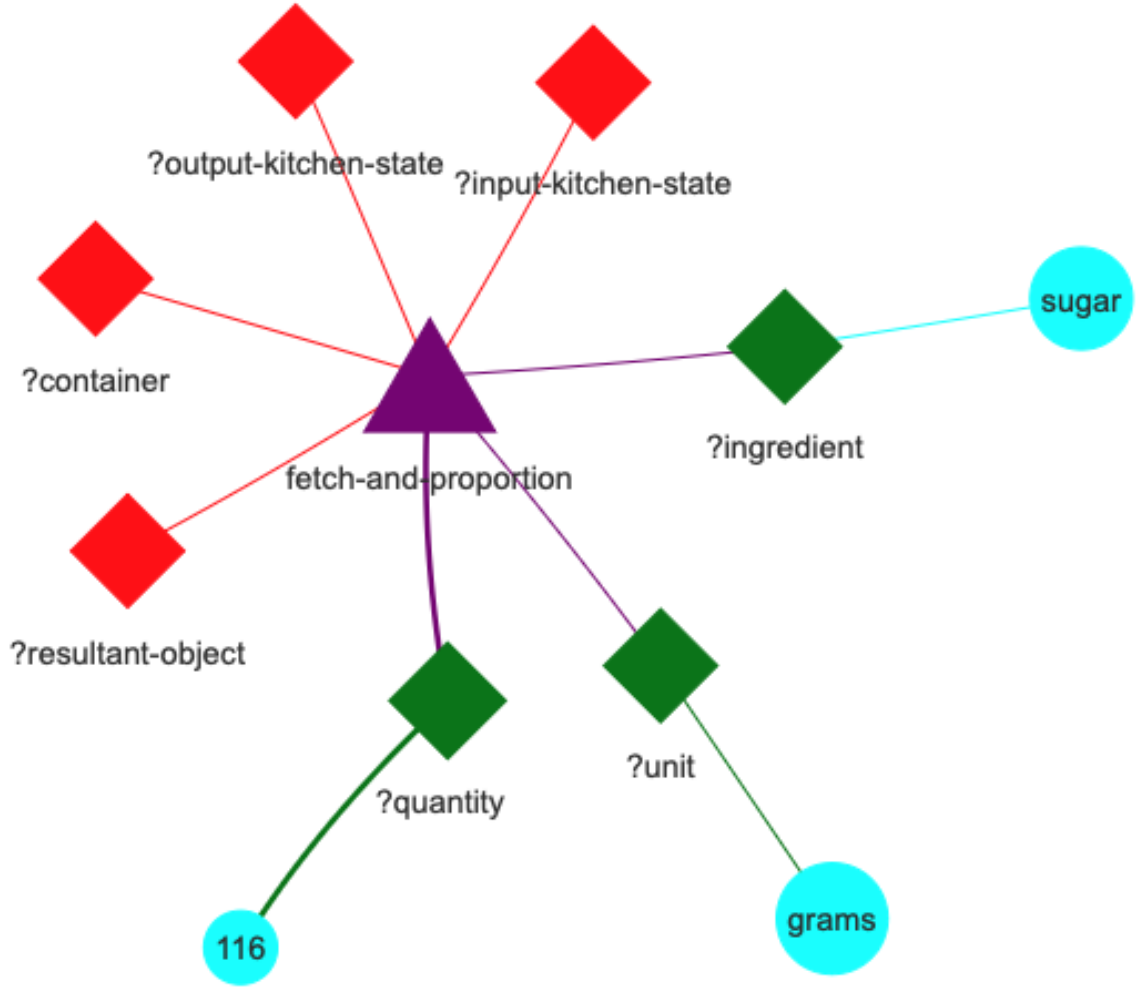


Figure 4.5: Parsing “116 grams sugar” leads to a partial robot plan that the cooking agent needs to complete into an executable one.

diamonds. The cooking agent thus needs to find fillers for those slots using different knowledge sources such as its Personal Dynamic Memory and mental simulation. To enable the agent to do so, we used the open-source software tool Incremental Recruitment Language (IRL) for representing, generating and executing robot plans [46, 44].

More specifically, we implemented a new representation language for cooking that includes 40 predefined cooking operations (called “primitives”) that encode meaning, temporality and dependencies. The IRL-system can then combine these primitives into graphs that represent complete recipe execution plans (ANONYMOUS, YEAR). Recurrent graphs can be automatically chunked and stored as composite operations for more efficient plan generation in the future, and users may extend the representation language with additional cooking operations.

As shown in Figure 4.3 plan execution relies on mental simulation, which is a distinct human capacity that allows us to project ourselves into hypothetical realities [61], typically in the form of narratives [16]. In our case study, we therefore included our own (qualitative) kitchen simulator as well as a quantitative simulator [38] for simulating cooking operations and their effects (see section 4.4.2).

One of the reasons for selecting the IRL-system for plan generation and plan execution is that it allows a *data flow* approach in which the handling of data adapts to which data is available at a given moment, as opposed to explicit control flow in which all operations need

to be ordered beforehand. For example, given an initial kitchen state, a food product and the unit of measurement, the operation *fetch-and-proportion* can compute the resultant object and output kitchen-state. Suppose however that a human chef has already taken a cup of 116 grams of sugar, then the agent could apply the same operation in a different direction for verifying whether the resultant object corresponds to what is written in the recipe, or for backtracking which actions the human user must have taken.

Data flow is important for handling the dynamic nature of a kitchen environment (where many things can go wrong) and for adapting to the user’s needs, who may use different variations of a recipe or who may have different preferences about which actions they like to perform themselves and which to delegate to their AI assistant. Moreover, data flow allows the final robot plan to be greatly optimized. For instance, the cooking agent does not need to wait until an operation such as *boil* is completely finished: the IRL-system will already provide a placeholder for the resultant object (including a timing for when it will be ready) so the cooking agent can already start planning and executing other task.

The result of plan generation and execution is a new node in the plot that the agent is constructing in its Personal Dynamic Memory, as shown in Figure 4.3. This new node includes an update of the kitchen-state and which entities are currently accessible. Indeed, the “meaning” of each recipe instruction is a small executable robot plan that is causally linked to the next one. The interlinked plans as a whole form a detailed and coherent robot execution plan for the whole recipe, which makes it possible to backtrack to earlier kitchen states whenever necessary.

4.4 Evaluation and Self-Assessment

In this section we describe the steps taken towards evaluation as well as first results. Moreover, we discuss how a cooking agent may reason about its own performance.

4.4.1 Integrated Narrative Networks

One of the challenges of narrative-based understanding is to monitor how different knowledge sources are integrated with each other, and whether the resulting plot offers a coherent and sensible content model. Crucially, the cooking agent itself should also have a way to monitor its own understanding process.

We are approaching this challenge using a new data structure called *Integrative Narrative Networks* (ANONYMOUS, YEAR). The key inspiration for such networks comes from the narratological concept of “narrative questions”, which are the questions that are raised in the audience’s mind by a narrative (or that an author wants to be raised). For instance, when a new character is introduced in a movie, this may raise the narrative question “who is this person?” Compelling narratives use such questions to connect plot points to each other and to keep the audience engaged, until narrative closure is reached – the point where all salient questions have been satisfactorily answered [13]. Likewise, we can frame understanding as a process in which an agent poses narrative questions to itself, and then searches for answers for those questions until it reaches narrative closure (or a conclusion).

The network in Figure 4.5 illustrates this idea. This network shows that the activation of the primitive *fetch-and-proportion* raises a number of questions (depicted as diamond-shaped nodes), such as which ingredient to fetch and what the resultant object will be. Some of these questions are immediately answered by parsing “116 grams sugar” (the green nodes), while other questions are still open (red). The agent now has to search for answers for those questions. Narrative questions and answers can be introduced by various knowledge sources, and this

process is continued until all salient questions have been satisfactorily answered. Figure 4.6 shows a complete Integrative Narrative Network, as built by the cooking agent, for the almond crescent cookies recipe.

Figure 4.7 shows some results of a first experiment in monitoring and measuring narrative-based understanding, described in more detail in (ANONYMOUS, YEAR). The black line shows the number of narrative questions that are raised as the cooking agent goes through the recipe. The coloured segments show how many questions were answered by different knowledge sources. From bottom to top, these are: language (blue), mental simulation (orange), ontology (green) and the discourse model/PDM (red).

Our current work focuses on how such Integrative Narrative Networks can be used for allowing agents to monitor and reason about their own understanding process, and to optimize the decisions that they make in the face of uncertainty.

4.4.2 Recipe Execution Benchmark

In order to evaluate our work as well as invite the research community to advance the field of grounded language understanding (and computational recipe understanding in particular), we have released a fully documented Recipe Execution Benchmark (ANONYMOUS, YEAR), which consists of the following components:

- A representation language for cooking (see section 4.3.2) that can express complete recipe execution plans. This representation language is independent from syntax or a particular natural language, so knowledge about syntax is not necessary for annotation.
- A test set of 30 English recipes with gold standard annotations. These recipes have been selected for the specific linguistic and extralinguistic challenges for recipe understanding.
- A qualitative kitchen simulator that is able to execute the recipe execution plans, and which returns both execution and evaluation results for further inspection.
- A suite of metrics that allow multiperspective estimates to optimize transferability to real-world utility. These consist of Smatch [11], goal-condition success, Dish Approximation Score, and Recipe Execution Time.

4.5 Conclusions and Future Work

This paper explored how narrative-based language understanding can be used for processing cooking recipes in a way that allows robots or artificial cooking agents to execute those recipes in a dynamic kitchen environment. Through a case study on an English recipe for almond crescent cookies, we have shown how the rich content models built during narrative-based understanding can be exploited for tackling the specific challenges of recipe language, such as resolving zero anaphora by keeping track of which entities are currently under the cooking agent’s attention. We have thereby shown how language processing can be embedded in a system for representing, generating and executing robot plans, coupled to a kitchen simulator. Moreover, we have proposed how narratives may offer a new framework for monitoring and measuring understanding.

Even though we have illustrated our approach through a concrete implementation and accompanying web demonstration, we hope to have convinced the reader that the framework of narrative-based understanding can be operationalized in a multitude of ways. To this end we

have published a Recipe Execution Benchmark for comparing different solutions. One key component of this benchmark is a new representation language for cooking, which allows recipes to be annotated in a syntax- and natural-language-independent fashion.

Our current and future work focuses on automatically learning computational construction grammars for recipe understanding in order to scale our approach. For the reasons detailed in section 4.2.2, we believe that Construction Grammar shows great promise to deal with the specific challenges of grounded language understanding, and computational recipe understanding in particular. Important breakthroughs in the automated learning of such grammars have recently been achieved in the domain of Visual-Question Answering [37], which requires a mapping from questions to visual queries in similar ways as recipe instructions map onto executable robot plans.

By mimicking the sense-making process of humans, narrative-based language understanding can become a key component in the development of human-centric AI systems. Such systems have many potential benefits for society, as they may interact with humans in more intuitive and meaningful ways.

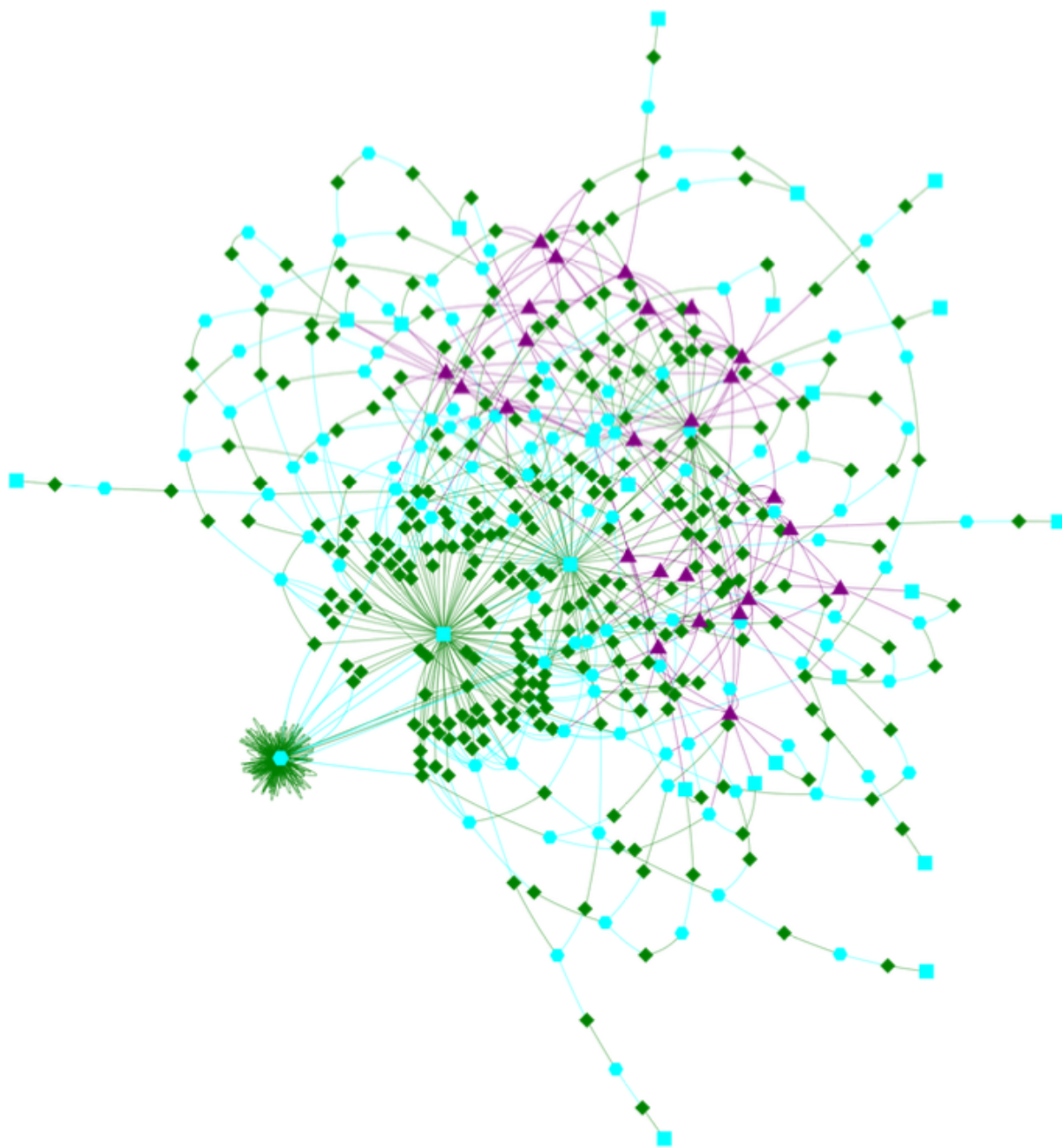


Figure 4.6: An Integrative Narrative Network for the almond crescent cookies recipe.

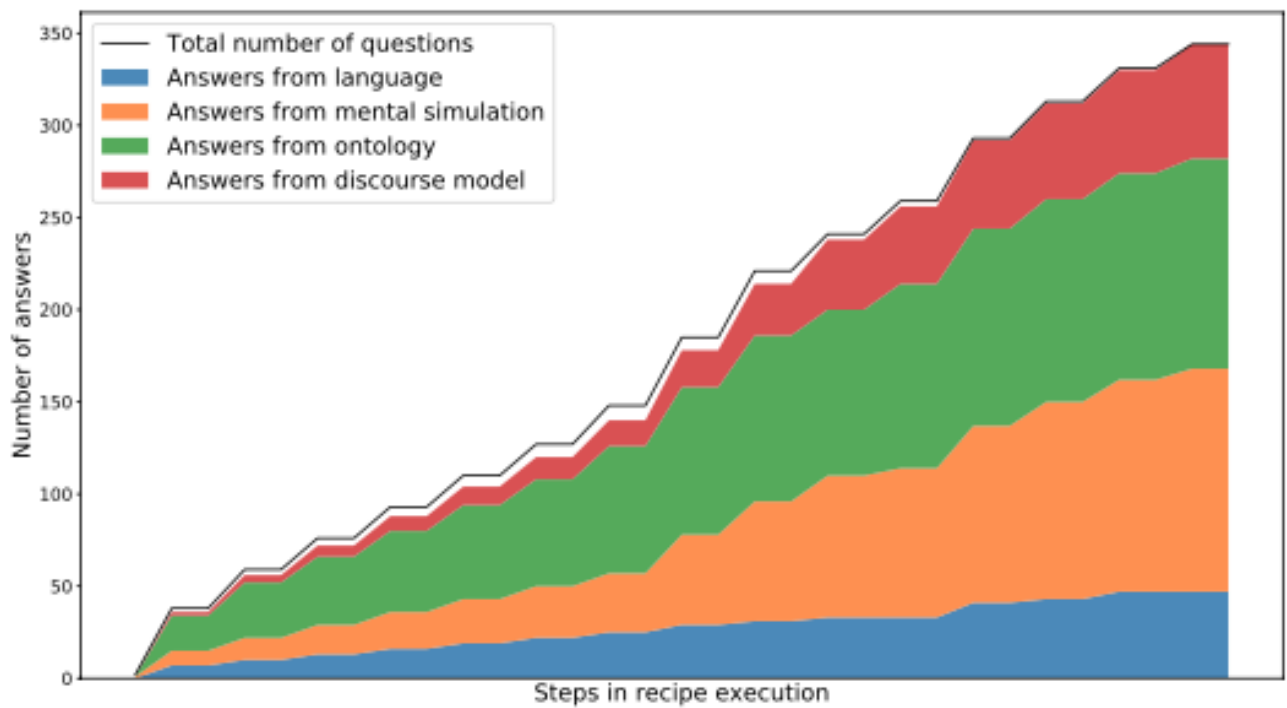


Figure 4.7: This Figure illustrates the number of narrative questions raised during recipe understanding and how many of them have been answered by which knowledge source.

Chapter 4: References

- [BB1] James F. Allen. *Natural Language Understanding*. Benjamin/Cummings, Redwood City, 2nd edition, 1994.
- [BB2] John Langshaw Austin. *How to Do Things with Words*. Clarendon Press, Oxford, 1962.
- [BB3] Mieke Bal. *Narratology: Introduction to the Theory of Narrative*. Toronto Univeristy Press, Toronto, 1985.
- [BB4] Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner, Dejan Pangercic, Thomas Rühr, and Moritz Tenorth. Robotic roommates making pancakes. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 529–536, 2011.
- [BB5] Ariel Bezaleli Mizrahi, Alexander “Zoonder” Lachnisch, and Amit Raphael Zoran. Digital gastronomy testcase: A complete pipeline of robotic induced dish variations. *International Journal of Gastronomy and Food Science*, 31:100625, 2023.
- [BB6] Inès Blin. Building narrative structures from knowledge graphs. In Paul Groth, Anisa Rula, Jodi Schneider, Ilaria Tiddi, Elena Simperl, Panos Alexopoulos, Rinke Hoekstra, Mehwish Alam, Anastasia Dimou, and Minna Tamper, editors, *The Semantic Web: ESWC 2022 Satellite Events. ESWC 2022*, volume 13384 of *Lecture Notes In Computer Science*, pages 234–251, Cham, 2022. Springer.
- [BB7] Hans C. Boas. *A Constructional Approach to Resultatives*. Stanford Monograph in Linguistics. CSLI, Stanford, 2003.
- [BB8] Mario Bollini, Stephanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. Interpreting and executing recipes with a cooking robot. *Experimental Robotics*, 88:481–495, 2013.
- [BB9] Jerome Bruner. The Narrative Construction of Reality. *Critical Inquiry*, 18(1):1–21, 1991. Publisher: The University of Chicago Press.
- [BB10] Alan Bundy and Lincoln Wallen. Procedural attachment. In Alan Bundy and Lincoln Wallen, editors, *Catalogue of Artificial Intelligence Tools*. Springer, Berlin/Heidelberg, 1984.
- [BB11] Shu Cai and Kevin Knight. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.
- [BB12] Patrizia Cani. De-constructing recipes: A constructionist comparative analysis of recipe texts. Master’s thesis, Università Ca’ Foscari Venezia, Venice, 2022.
- [BB13] Noël Carroll. Narrative closure. *Philosophical Studies*, 135(1):1–15, 2007.

- [BB14] Colleen Cotter. Claiming a piece of the pie: How the language of recipes defines community. In Anne L. Bower, editor, *Recipes for Reading: Community Cookbooks, Stories, Histories*, pages 51–72. University of Massachusetts Press, Amherst, 1997.
- [BB15] Regine Eckardt. Truth conditional semantics. In *Meaning Change in Grammaticalization: An Enquiry into Semantic Reanalysis*, pages 59–90. Oxford University Press, 2006.
- [BB16] Jennifer Edson Escalas. Imagine yourself in the product: Mental simulation, narrative transportation, and persuasion. *Journal of Advertising*, 33(2):37–48, 2004.
- [BB17] Charles J. Fillmore. The Mechanisms of “Construction Grammar”. In *Proceedings of the Fourteenth Annual Meeting of the Berkeley Linguistics Society*, pages 35–55, Berkeley CA, 1988. Berkeley Linguistics Society.
- [BB18] Mirjam Fried. Discourse-referential patterns as a network of grammatical constructions. *Constructions and Frames*, 13(1):21–54, 2021. ISBN: 1876-1933 Publisher: John Benjamins Type: <https://doi.org/10.1075/cf.00046.fri>.
- [BB19] Mirjam Fried and Jan-Ola Östman. Construction Grammar: A Thumbnail Sketch. In Mirjam Fried and Jan-Ola Östman, editors, *Construction Grammar in a Cross-Language Perspective*, pages 11–86. John Benjamins, Amsterdam, 2004.
- [BB20] Cornelia Gerhardt, Maximiliane Frobenius, and Susanne Ley, editors. *Culinary Linguistics: The Chef’s Special*, volume 10 of *Culture and Language Use*. John Benjamins, Amsterdam, 2013.
- [BB21] Adele E. Goldberg. Constructions: A New Theoretical Approach to Language. *Trends in Cognitive Science*, 7(5):219–224, May 2003.
- [BB22] Adele E. Goldberg and Ray Jackendoff. The English Resultative as a Family of Constructions. *Language*, 80(3):532–568, 2004.
- [BB23] Stevan Harnad. The Symbol Grounding Problem. *Physica D*, 42:335–346, 1990.
- [BB24] Caleb Harper and Mario Siller. Openag: A globally distributed network of food computing. *IEEE Pervasive Computing*, 14(4):24–27, 2015.
- [BB25] Peter Hart and Nils Nilsson. Shakey: An experiment in robot planning and learning. Movie, 1972.
- [BB26] Steven Haussmann, Oshani Seneviratne, Yu Chen, Yarden Ne’eman, James Codella, Ching-Hua Chen, Deborah L. McGuinness, and Mohammed J. Zaki. FoodKG: a semantics-driven knowledge graph for food recommendation. In Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtěch Svátek, Isabel Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon, editors, *The Semantic Web – ISWC 2019*, pages 146–162, Cham, 2019. Springer.
- [BB27] Hajira Jabeen, Jonas Weinz, and Jens Lehmann. Autochef: Automated generation of cooking recipes. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–7, 2020.
- [BB28] Yiwei Jiang, Klim Zaporozhets, Johannes Deleu, Thomas Demeester, and Chris Davelder. Recipe instruction semantics corpus (RISeC): Resolving semantic structure and zero anaphora in recipes. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint*

Conference on Natural Language Processing, pages 821–826, Suzhou, China, December 2020. Association for Computational Linguistics.

- [BB29] Abdullah Faiz Ur Rahman Khilji, Riyanka Manna, Sahinur Rahman Laskar, Partha Pakray, Dipankar Das, Sivaji Bandyopadhyay, and Alexander Gelbukh. CookingQA: Answering Questions and Recommending Recipes Based on Ingredients. *Arabian Journal for Science and Engineering*, 46(4):3701–3712, April 2021.
- [BB30] Hanna Kicherer, Marcel Dittrich, Lukas Grebe, Christian Scheible, and Roman Klinger. What you use, not what you do: Automatic classification and similarity detection of recipes. *Data & Knowledge Engineering*, 117:252–263, 2018.
- [BB31] Hadas Kress-Gazit, Morteza Lahijanian, and Vasumathi Raman. Synthesis for robots: Guarantees and feedback for robot behavior. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:211–236, 2018.
- [BB32] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.
- [BB33] Riyanka Manna, Dipankar Das, and Alexander Gelbukh. Information retrieval-based question answering system on foods and recipes. In Lourdes Martínez-Villaseñor, Oscar Herrera-Alcántara, Hiram Ponce, and Félix A. Castro-Espinoza, editors, *Advances in Computational Intelligence*, pages 260–270, Cham, 2020. Springer International Publishing.
- [BB34] Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.
- [BB35] Weiqing Min, Shuqiang Jiang, Linhu Liu, Yong Rui, and Ramesh Jain. A survey on food computing. *ACM Computing Surveys*, 52(5), sep 2019.
- [BB36] Nadia A. Najjar and David C. Wilson. Computer cooking contest 2017 – preface. In Antonio A. Sanchez-Ruiz and Anders Kofod-Petersen, editors, *Proceedings of ICCBR 2017 Workshops (CAW, CBRDL, PO-CBR), Doctoral Consortium, and Competitions co-located with the 25th International Conference on Case-Based Reasoning (ICCBR 2017)*, pages 224–227, Trondheim, 2017. CEUR Workshop Proceedings.
- [BB37] Jens Nevens, Jonas Doumen, Paul Van Eecke, and Katrien Beuls. Language acquisition through intention reading and pattern finding. In N. Calzolari and C.-R. Huang, editors, *Proceedings of the 29th International Conference on Computational Linguistics*, pages 15–25. International Committee on Computational Linguistics, 2022.
- [BB38] Mihai Pomarlan. AbeSim. https://github.com/mpomarlan/abe_sim, 2021.
- [BB39] Gorjan Popovski, Barbara Koroušić Seljak, and Tome Eftimov. FoodBase corpus: a new resource of annotated food entities. *Database*, 2019:baz121, 11 2019.
- [BB40] John R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, 1969.
- [BB41] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10737–10746, 2020.

- [BB42] Neva Smith and King-Ip Lin. Automatic information extraction from the web: Case study with recipes. In *Proceedings of the 50th Annual Southeast Regional Conference, ACM-SE '12*, pages 369–370, New York, NY, USA, 2012. Association for Computing Machinery.
- [BB43] Dan Sperber and Deirdre Wilson. *Relevance: Communication and Cognition*. Basil Blackwell, Oxford, 1986.
- [BB44] Michael Spranger, Simon Pauw, Martin Loetzsch, and Luc Steels. Open-ended Procedural Semantics. In Luc Steels and Manfred Hild, editors, *Language Grounding in Robots*, pages 153–172. Springer, New York, 2012.
- [BB45] Luc Steels. Procedural attachment. Technical Report AIM-543, MIT, Cambridge MA, 1979.
- [BB46] Luc Steels. The Emergence of Grammar in Communicating Autonomous Robotic Agents. In W. Horn, editor, *ECAI 2000: Proceedings of the 14th European Conference on Artificial Life*, pages 764–769, Amsterdam, August 2000. IOS Publishing.
- [BB47] Luc Steels. Constructivist Development of Grounded Construction Grammars. In Walter Daelemans and Marilyn Walker, editors, *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 9–19, Barcelona, 2004. Association for Computational Linguistic Conference.
- [BB48] Luc Steels. Personal dynamic memories are necessary to deal with meaning and understanding in human-centric ai. In Alessandro Saffiotti, Luciano Serafini, and Paul Lukowicz, editors, *Proceedings of the First International Workshop on New Foundations for Human-Centered AI (NeHuAI) co-located with 24th European Conference on Artificial Intelligence (ECAI 2020)*, volume 2659, pages 11–16, Santiago de Compostella, 2020. CEUR Workshop Proceedings.
- [BB49] Luc Steels, editor. *Foundations for Meaning and Understanding in Human-Centric AI*. Venice International University, Venice, 2022.
- [BB50] Luc Steels and Joachim De Beule. Unify and Merge in Fluid Construction Grammar. In P. Vogt, Y. Sugita, E. Tuci, and C. Nehaniv, editors, *Symbol Grounding and Beyond: Proceedings of the Third International Workshop on the Emergence and Evolution of Linguistic Commun*, LNAI 4211, pages 197–223. Springer-Verlag, Berlin, 2006.
- [BB51] Yuta Sugiura, Daisuke Sakamoto, Anusha Withana, Masahiko Inami, and Takeo Igarashi. Cooking with robots: Designing a household system working in open environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2427–2430, New York, 2010. Association for Computing Machinery.
- [BB52] Nicolas Szilas. Towards Narrative-Based Knowledge Representation in Cognitive Systems. In Mark A. Finlayson, Ben Miller, Antonio Lieto, and Remi Ronfard, editors, *6th Workshop on Computational Models of Narrative (CMN 2015)*, volume 45 of *OpenAccess Series in Informatics (OASIs)*, pages 133–141, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [BB53] Stefanie Tellex, Nakul Gopalan, Hadas Kress-Gazit, and Cynthia Matuszek. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):25–55, 2020.
- [BB54] Moritz Tenorth, Daniel Nyga, and Michael Beetz. Understanding and executing in-

- structions for everyday manipulation tasks from the world wide web. In *2010 IEEE International Conference on Robotics and Automation*, pages 1486–1491, 2010.
- [BB55] Yijun Tian, Chuxu Zhang, Zhichun Guo, Chao Huang, Ronald Metoyer, and Nitesh V. Chawla. Reciperec: A heterogeneous graph learning model for recipe recommendation. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3466–3472. International Joint Conferences on Artificial Intelligence Organization, 7 2022. Main Track.
- [BB56] Yijun Tian, Chuxu Zhang, Ronald Metoyer, and Nitesh V. Chawla. Recipe representation learning with networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM '21*, pages 1824–1833, New York, NY, USA, 2021. Association for Computing Machinery.
- [BB57] Paul Van Eecke, Jens Nevens, and Katrien Beuls. Neural heuristics for scaling constructional language processing. *Journal of Language Modelling*, 10(2):287–314, 2022.
- [BB58] Paul Van Eecke, Lara Verheyen, Tom Willaert, and Katrien Beuls. The Candide model: How narratives emerge where observations meet beliefs. In *Proceedings of the The 5th Workshop on Narrative Understanding*, pages 48–57, Toronto, Canada, July 2023. Association for Computational Linguistics.
- [BB59] Remi van Trijp, Katrien Beuls, and Paul Van Eecke. The FCG Editor: An innovative environment for engineering computational construction grammars. *PLOS ONE*, 17(6):e0269708, June 2022. Publisher: Public Library of Science.
- [BB60] Hao Wang, Guosheng Lin, Steven C. H. Hoi, and Chunyan Miao. Learning structural representations for recipe generation and food retrieval, 2022.
- [BB61] Adam Waytz, Hal E. Hershfield, and Diana I. Tamir. Mental simulation and meaning in life. *Journal of Personality and Social Psychology*, 108(2):336–355, 2015.
- [BB62] Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. AI Technical Reports AITR-235, MIT, Cambridge MA, 1971.
- [BB63] Ludwig Wittgenstein. *Philosophical Investigations*. Wiley-Blackwell, Oxford, 1953.
- [BB64] Semih Yagcioglu, Aykut Erdem, Erkut Erdem, and Nazli Ikizler-Cinbis. RecipeQA: A challenge dataset for multimodal comprehension of cooking recipes. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1358–1368, Brussels, Belgium, 2018. Association for Computational Linguistics.
- [BB65] Amit Zoran. Cooking with computers: The vision of digital gastronomy [point of view]. *Proceedings of the IEEE*, 107(8):1467–1473, 2019.

Chapter 5

A Cooking Benchmark for Ingredient Substitution

Nikolaos Kondylidis, Ilaria Tidli, Annette ten Teije

Vrije Universiteit Amsterdam

Abstract

Keywords

5.1 Introduction

Food understanding [2] is gaining an increasing amount of popularity in AI, as it can help to study the cooking process while also the nutritional aspects of a meal. In this section we will focus on the task of recipe redesign and particularly on ingredient substitution. We justify the importance of the task and provide a summary of the chapter. We then outline related studies in Section 5.2, describe how human-centric understanding can be established by interacting with a chef in Section 5.3 and present our proposed agent policies in Section 5.4. Then, we set out our experimental design that allows the evaluation of our approach in Section 5.5. Finally, we draw our conclusions and directions of future work in Section 5.6.

The task of substituting ingredients in recipes has been gaining a lot of attention within the field of AI as it has direct societal benefit being a Human-centric task. Human-centric AI focuses on collaborating with humans, enhancing human capabilities, and empowering humans to better achieve their goals [8]. Firstly, ingredient substitution can help one to move to a healthier diet, without requiring them to move away from their favorite meals. Secondly, it can allow individuals with allergies or other restricting factors, i.e. culture, to enjoy dishes that they could not before. Thirdly, it enables the preparation of recipes that contain inaccessible or expensive ingredients. Last but not least, one can reduce the consumption of ingredients that have a big environmental impact.

Ingredient substitution, being human-centric, falls under the category of cultural or subjective tasks, where people are expected to have diverse opinions or preferences. Additionally, different chefs may not only disagree on the ingredients that can be used as a substitution for a specific case, but also on their reasoning behind their decision. It can be due to ingredient flavor similarity, the type of recipe, or may be contingent on cultural or traditional influences. In such an intuitive and creative process, it can be difficult to understand why a chef is making a specific proposal. One way of approaching this problem would be to use Knowledge Representation

(KR), but it is challenging to find an expert that outlines the rules. Another way would be by using Machine Learning (ML), but this is a data-hungry approach, requiring very large amounts of ingredient substitution examples. One would have to ask a chef to provide them, or could use existing datasets. The former case would be very expensive, requiring a large amount of time from a chef. The latter case is based on data gathered from several sources, representing multiple cooking styles. A ML approach would be unable to capture underrepresented or diverse cooking styles, since its nature is to focus on performing well on the most common cases.

Ingredient substitution is a challenging task that requires a good understanding of the ingredients’ interdependency and the cultural norms that surround it. Our premise is that ingredient substitution falls under the category of tasks that we perform intuitively. For this reason, our hypothesis is that an agent can best develop such human-like understanding on substituting ingredients by asking a chef to provide a few substitution examples. Specifically, we aim to design an agent that learns from grounded ingredient substitution examples provided by a chef, as a mean to establish shared task-oriented understanding [5]. The agent is then utilizing different knowledge sources to i) interpret the provided examples, i.e. annotate them with additional information, ii) understand them, i.e. translate them into personalized semantically annotated experiences, and iii) learn from them, i.e. update its understanding. The aforementioned knowledge sources will consist of categories used by humans, like for example “low fat”, or “vegan”. At the same time, the agent will be designed to use these categories to understand why or when an ingredient can substitute another, developing a human-centric and human-like understanding of the task.

As described earlier, the agent is learning over a set of substitution examples provided by a chef. It is vital that the agent is able to learn efficiently from these experiences, aiming to minimize the required human interaction time, while at the same time achieving high task performance. To that end, the agent’s **active learning policy** is responsible for selecting the order of the examples over which the agent should learn, while its **learning policy** is tasked with updating the agent’s understanding of the task. For the former, we use an active learning method that utilizes the experiences memorized by the agent and actively decide which example would be the most informative to interact over next. Regarding the learning policy, we deploy a statistical method.

We extend existing datasets and benchmarks in order to test the proposed agent policies. In more detail, we perform Entity Linking (EL) on the Recipe1MSubs ingredient substitution benchmark to represent it in the form of a Knowledge Graph (KG). This allows us to further relate the ingredients involved with additional knowledge sources containing ingredient categorical information or nutritional data. Then, we can perform experiments that allow us to evaluate the proposed agent policies. While the main concern is the successful performance of the task, it must be evaluated with respect to the amount of input it required from the chef. Therefore, the agent’s performance is presented as a relation of task performance and number of examples provided.

5.2 Related Work

The rising popularity around the domain of food understanding has resulted in a number of studies that have produced a variety of data collections and methods. We will now review the ones that are related to our work. First, studies that have produced datasets, ontologies and KGs will be presented. Following, different approaches to the task of ingredient substitution will be presented. Finally, we will explain why these methods cannot be applied under our premise of learning via interacting with a chef.

5.2.1 Datasets

The Recipe1M dataset Recipe1M [7] is a landmark dataset in the domain of food understanding due to its size. It consists of over a million recipes scrapped from more than a dozen popular websites. Each recipe comes with i) a title, ii) a sequence of cooking instructions, iii) a list of ingredients and iv) photographs of the resulting dish. Besides the recipe’s title, both the cooking instructions and the ingredients are represented in natural language. Each recipe ingredient comes with a quantity and a unit of measurement. The total 1,029,720 recipes were split to 720,639, 155,036, and 154,045 sets for training, validation, and testing, respectively. From the recipes’ websites, a total of 716,480 images were related to some recipes. With the use of a search engine, the number of collected images was increased to 13,735,679, producing the Recipe1M+ dataset. Last but not least, the authors retrieved the nutritional information for some ingredients, using a published nutrient database¹, put together by the United States Department of Agriculture (USDA).

The Recipe1MSubs dataset [2] extends the Recipe1M with ingredient substitutions, for 70,520 of its recipes. The substitutions were crawled from the user provided comments section written on the original recipe website. Inline with the data format of Recipe1M, the authors represent both the recipes and the ingredients in natural language. Following the dataset splits of the recipes in the original Recipe1M dataset, the Recipe1MSubs contains 49,044, 10,729, and 10,747 ingredient substitution samples for the training, validation, and testing splits, respectively. Finally, the authors link ingredients with the FlavorGraph dataset [9], which is introduced in the next paragraph.

FlavorGraph [9] contains co-occurrence and molecular information for 6,653 ingredients that is represented in a graph format. Specifically, it contains ingredient co-occurrence information based on the times two ingredients are found in the same recipe, using the recipes of Recipe1M. In addition, ingredients are linked to 1,646 chemical compounds like for example “Nonanoic Acid”. This information is derived from FlavorDB [3] and HyperFoods [12]. The authors of the study put together this information in order to generate reusable ingredient embeddings that reflect their compatibility or similarity.

The Food.com dataset [6] comprises over 180 thousand recipe texts and at least 700 thousand user reviews, from a period of 18 years. As the name suggests, they were scrapped from the food.com website; formerly GeniusKitchen.com. The recipes were selected to have at least 3 steps and to involve from 4 to 20 ingredients. The total number of unique ingredients in the dataset is 13,000.

5.2.2 Ontologies and Knowledge Graphs

FoodOn [1] is a consortium-driven project aiming to create a unifying ontology defining concepts around food. In practice, it is an umbrella ontology, unifying and connecting a number of ontologies that serve different purposes. This way, FoodOn is able to describe a very large spectrum of concepts, envisioning to represent the complete trajectory of a food product from nature to dish. For example, the ontology is able to describe ingredient origins in terms of environment, geography, anatomy, and evolution. Furthermore, within FoodOn there are concepts that describe the gathering, storing and transporting processes. Additionally, kitchenware, cooking actions and categories of meals are also included. FoodOn also contains nutritional descriptions and links to diseases.

¹<https://data.nal.usda.gov/dataset/composition-foods-raw-processed-prepared-usda-national-nutrient-database-standard-reference-release-27>

FoodKG [4] is a KG representing the recipes of Recipe1M in Semantic Web (SW) techniques, following the nanopublication model. The authors of the study link the ingredients to FoodOn concepts and ingredients within the USDA aforementioned dataset. The former links allow the characterization of ingredients with categories like “animal product” or “citrus fruit juice”. The latter provides the KG with the nutritional information of some ingredients. Information regarding ingredients’ proportions per recipe are also inside the KG. Additionally, ingredients and recipes are striped from their natural language form and are represented using Internationalized Resource Identifiers (IRI)s. IRIs come from the SW domain and are in practice web addresses used to accurately represent distinct entities and concepts on the web. They provide a common vocabulary of meanings that everyone can use for reference, enhancing uniformity of representation and reducing disambiguation. The creators of FoodKG decide to not include the recipe execution actions in the produced representation. Lastly, the nutritional information of some ingredients is already incorporated in FoodKG, since it is already included in Recipe1M.

5.2.3 Tasks, Benchmarks, and Approaches on Ingredient Substitution

The Recipe1MSubs benchmark assumes that for each recipe R , there is a source ingredient ing_s that we want to replace and a target ingredient ing_t that can be used as a substitute. The task is to recommend the target ingredient ing_t , provided the recipe R and the source ingredient ing_s . The benchmark follows two task assumptions that we adopt. First, the substitutions are *directed*, meaning that although the target ingredient can replace the source ingredient, the reverse does not necessarily hold. Second, the substitutions are *recipe specific*, restricting us from assuming that a target ingredient can replace a source ingredient in any recipe, if it can do so in one. Since the task is to recommend the target ingredient for a given substitution case, the authors propose a model that ranks all ingredients and use the rank of the target ingredient to measure its recommendation performance. It should be noted that the original vocabulary of ingredients is shrunk to map the one of FlavorGraph [9], resulting to a vocabulary size of 6,653 ingredients. Last but not least, when different target substitutions are possible for the same source ingredient and recipe, the authors of the benchmark accept the highest ranked correct one, among the possible options.

The authors of Recipe1MSubs put to the test a Neural Network (NN) [2]. In more detail, the applied NN consists of an ingredient encoder, a context encoder, and an ingredient substitution decoder. The ingredient encoder is a graph isomorphism network [13] that outputs node embeddings, representing ingredients. The context encoder is simply averaging the output ingredient embeddings of a provided recipe. The ingredient substitution decoder is generating substitution scores for the ingredients in the corpus. The complete network is receiving as input the ingredients of a recipe and explicitly the ingredient that needs to be substituted, producing candidate ingredient substitution scores. GISMo is trained in an End-to-End setting, for the objective of maximizing the score of the correct target substitution ingredient, in contrast to all other ingredients. This study reports very large performance improvement over the other presented NN-based approaches, but small improvements compared to simpler statistical approaches, albeit their minimal computational training demands. These findings suggest that the Recipe1MSubs benchmark might not require a computationally expensive and complex NN for achieving high performance. Additionally, the suggested approach requires very large substitution examples for training. In contrast, our approach is designed for scenarios where examples are few and difficult to collect.

The next mentioned study² is a good example of combining symbolic and sub-symbolic tech-

²<https://foodkg.github.io/subs.html>

niques for the task of ingredient substitution. The authors design the Diet-Improvement Ingredient Substitution Heuristic (DIISH). Its sub-symbolic part functions using word-based ingredient embeddings, some of which were trained on the lexical part of the Recipe1M benchmark. The symbolic part estimates ingredient pairing and recipe context similarities using information from FoodKG to further estimate ingredient similarity. The authors report that their heuristic exhibits task performance improvement over the individual heuristics that it consists of. The ingredient substitution data used for the experiments were crawled from a cooking ingredient focused website and the comments of the food.com dataset. Unfortunately, the updated terms and conditions of the website do not allow us to replicate the generation of the substitution benchmark that was used in this study. Limitations of this work include the fact that the sub-symbolic part is trained in an unsupervised manner and does not use the ingredient substitutions as a training signal. This lack of task oriented training restricts the proposed heuristic from mimicking specific substitution styles of a chef, that our approach aims to capture.

The work of [10] uses graph node embeddings to represent both ingredient and cooking actions, allowing them to go beyond single ingredient substitution. In fact, this approach can substitute ingredients and cooking actions at the same time. First, the authors translate the recipe instructions text into an acyclic flow graph that uses nodes and edges to represent ingredients and cooking actions, respectively. Both the nodes and the edges are represented using trainable embeddings. During this process, only verbs and ingredients that could be linked to FoodOn were represented, ensuring their task relevance. The FoodOn classes that were linked to ingredients provided additional information during the training of the method. Then, the process of applying a cooking action to an ingredient is defined to be represented as the summation of their corresponding embeddings. This allows for the representation of any possible recipe, as a sequence of actions applied on ingredients. It does so while only requiring the trainable representation of ingredients and cooking actions, and not the explicit representation of all combinations. Such a compositional approach makes it possible to i) represent unforeseen combinations and ii) estimate the semantic distance between any two recipes. It should be noted that the embeddings were actually not trained using ingredient substitution as a signal, nor they were evaluated for this task. Instead, they were trained in an unsupervised manner for the purpose of identifying a concealed ingredient of a recipe.

5.2.4 Active Learning

Active Learning (AL) is an area of ML where the model has an active participation in its learning process. We can describe the training of a model as the process that alters it in a way that it can produce the most correct output for any given input. The model alterations are performed according to training samples. In traditional, passive learning approaches, training samples are given to the model in a random order. In contrast, AL optimizes the learning process by strategically choosing the most informative samples to learn from next. In AL algorithms, the strategical selection of training samples is dictated by the “Query strategy”, that ranks available samples using a “utility function” according to their estimated informativeness. The estimated informativeness of a sample is only based on its input, since its output is not known to the model yet. AL allows for more sample-efficient model training [11], making it an ideal candidate for applications where data labeling is resource-intensive, like our scenario. Thus, an AL policy will also be incorporated by our agent, described in Section 5.4.2.

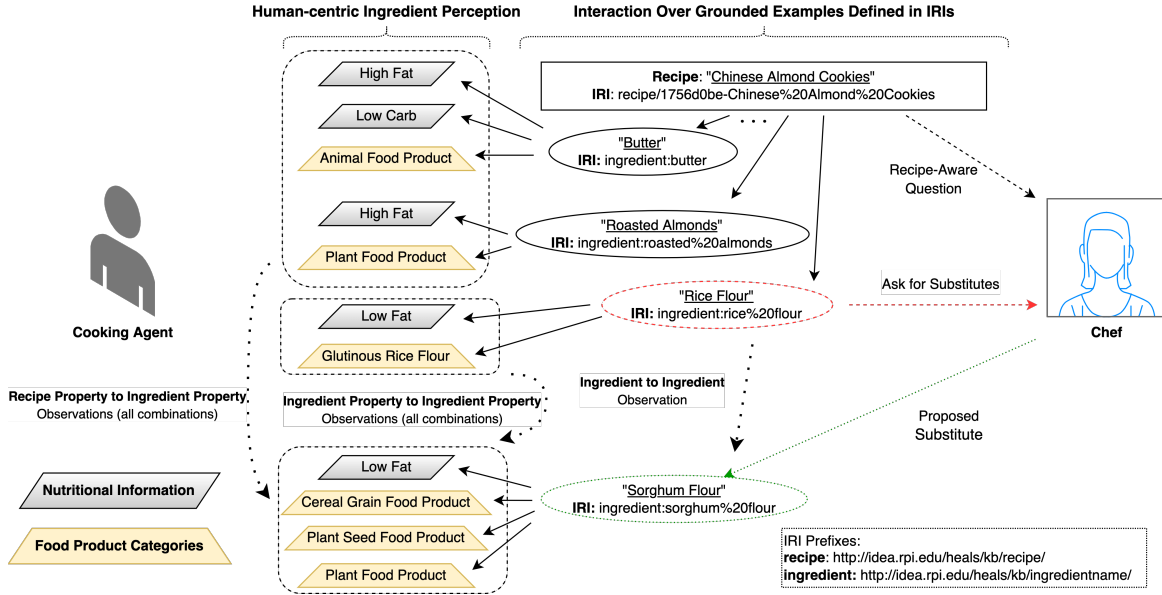


Figure 5.1: A depiction of a recipe-aware ingredient substitution example from our training set. The middle of the figure shows both the ingredients and the recipe, represented using Internationalized Resource Identifiers (IRI)s, ensuring disambiguous interaction between the chef and the cooking agent. The agent asks the chef how to substitute a particular ingredient, i.e. “Rice Flour”, for this particular recipe. It then receives the chef’s response, i.e. “Sorghum Flour”. Furthermore, on the left side we see the agent’s human-centric perception of the ingredients, i.e. “Low Fat”, “Plant Food Product”, which enables the agent to develop a deeper and human-like understanding of the ingredients’ interplay. Food product categories and nutritional information come from different knowledge sources and for this reason are visualized differently. Finally, we visualize the three types of ingredient substitution observations, i.e. “Ingredient to Ingredient”, “Ingredient Property to Ingredient Property” and “Recipe Property to Ingredient Property”, that agent memorizes from this interaction experience.

5.2.5 A Novel Task Definition is Needed for a Human-centric Approach

Our premise suggests that ingredient substitution should be incrementally learned over a few interactions with a chef. This is to tackle the challenges of having to explicitly represent a subjective and intuitive humane understanding for this task, that would be required for a KR approach. Since the input would be provided by a domain expert, i.e. a chef, it is important to minimize their time and provided input, deeming the data-hungry ML approaches unfit for such an approach. It is therefore our belief that the task should be approached and evaluated differently, leading also to the development of agent policies that learn from interaction experiences with humans experts. Based on the presented related work, we aim to combine AL with simple statistical learning methods, aiming to design agent policies that learn effectively and efficiently. In the next section, we dive into the details of our suggested approach to the task, while also the performed necessary steps that allow its computational definition and experimental study.

5.3 Establishing Human-centric Understanding via Interactive Learning

In this section, we will describe the virtual environment where the agent can learn to understand how to perform ingredient substitution in a human-centric manner, while also the process of

setting it up. We first define the ingredient substitution task computationally and how we ensure correct identification of the recipes and ingredients between the agent and the chef via grounded communication. Then, we present how the agent and the chef can establish a shared understanding of the ingredient substitution task via grounded interaction, following the guidelines of [5]. In more detail, we describe the interaction setting that allows us to artificially represent the chef’s input as to update and evaluate the agent’s understanding of the task. Finally, we detail how the agent’s perception is designed, so to perceive the ingredients in a human-centric manner using different knowledge sources. Figure 5.1 illustrates the chef providing a substitution example for a specific recipe to the cooking agent. Please also note in the figure how the interaction is taking place over IRIs, ensuring unambiguous communication while also a paradigm of the agent’s ingredient perception in human-centric terms. The illustrated substitution example is a result of our entity linking and grounding process, extending earlier work. The recipe and the substitution suggestion were crawled from a cooking related website³ and is part of the Recipe1MSubs training set.

5.3.1 Defining and Grounding the Ingredient Substitution Task

We use the Recipe1MSubs benchmark that extends the data of Recipe1M with ingredient substitutions, provided also in natural language form. Textual representations can cause miscommunication between the agent and the chef, due to their ambiguous nature. To prevent this, we map the recipes, the source ingredients, and the target ingredients of Recipe1MSubs to IRIs, using EL. We call the process of representing the Recipe1MSubs using IRIs as “grounding”, because its product can be used as a common ground for disambiguous communication. FoodKG already represents the original recipes and ingredients of Recipe1M in IRIs. We therefore had to additionally link the recipes and ingredients between Recipe1MSubs and FoodKG.

The recipes can be identified with their recipe ID, found in Recipe1M and Recipe1MSubs, but also from the URL they were crawled from, included in Recipe1M and FoodKG. We can therefore use Recipe1M as a linking node to relate recipes between Recipe1MSubs and FoodKG. Linking the ingredients was performed through lexical matching. Provided an ingredient substitution sample from Recipe1MSubs, we had to identify the source ingredient with one of the ingredients of the corresponding recipe registered in FoodKG, and identify the target ingredient with any ingredient in FoodKG. The former was necessary, to be able to know which ingredient needs to be removed from the original recipe. Furthermore, FoodKG can have more than one recipe entry from the same crawled URL, requiring us to match the source ingredient, to make sure we have linked the correct recipe as well. The lexical matching of the ingredients from Recipe1MSubs to FoodKG is performed as follows. We first remove any quoting characters and apostrophes from the name. Then we try to match the ingredient either directly with the IRI name, or with the annotated name of that IRI, i.e. annotated with the *rdfs:label* property. When attempting to match with an IRI, we replace all ‘_’ characters with ‘%20’ or ‘%20-%20’, while when matching with a name, we replace them with the *Space* ‘ ’ character or ‘ - ’. Last but not least, in case of any ‘_’ character, we treat them as separators of tokens, and we try matching all combinations of capitalizing the first letter of every token separately. At the end, each sample is represented as i) a set of ingredient IRIs for the recipe, ii) the IRI of the source ingredient and iii) the IRI of the target ingredient. Following the aforementioned process, we identify both ingredients of a substitution in 38,142, 8,451 and 8,582 samples of the training, validation, and testing splits, respectively. Having mapped almost 80% of the substitutions in IRIs, we can already start developing methods and perform experiments. We aim to link the remaining substitutions of Recipe1MSubs during our next steps.

³<http://www.food.com/recipe/chinese-almond-cookies-329688>

The Recipe1MSubs benchmark comes with 6.653 substitution vocabulary size. We, on the other hand, use the complete set of ingredients found in FoodKG, of size 16,780. Furthermore, we require the agent to find the target ingredient for each sample, not counting other correct substitution as valid, in contrast to the approach of the authors of Recipe1MSubs. This is done, so that the agent has to learn all possible correct substitutions to record high performance, not only one of them. We also approach this task as an information retrieval task and evaluate the agent’s task performance based on the ranking of the target ingredient among the suggested substitutions.

5.3.2 Efficiently Teaching and Evaluating the Agent via Interaction

We assume that the best way for an agent to learn how to perform ingredient substitution is to figure out the “why” by itself, from examples provided by a chef. Therefore, we do not aim to design an algorithm that independently learns how to perform the task, but to design an agent policy that is able to pick up such knowledge by efficiently interacting with a chef. We will now describe this interactive learning process and how to evaluate its performance in terms of efficiency.

The chef is providing ingredient substitution examples to the agent, and the latter needs to learn from them and generalize the gained knowledge to other recipes and other ingredients. The agent has access to combinations of recipes and their corresponding source ingredients of the training set. Provided these, it can decide which target ingredient of a combination should the chef reveal next. In case of multiple possible substitutions, the chef reveals one of them, randomly.

It is vital to minimize the interaction time of the domain expert, i.e. the chef. To that end, we evaluate the task performance of the agent’s learning policies with respect to the number of examples the chef has provided until that point. The **number of examples provided** is used as an evaluation metric of the agent’s learning efficiency. Together with the task performance evaluation metrics, it allows us to depict both how well the agent learned the task, but also in what cost. Therefore, every N examples provided by the chef, we evaluate the agent’s task performance on the validation set.

5.3.3 Providing the Agent with a Human-centric Ingredient Perception

The ingredient substitution examples that are provided to the agent consist of IRIs, representing ingredient. Treating recipes as sets of ingredients and at the same time representing ingredients merely using IRIs, while not knowing anything more about them besides ingredient co-occurrences and replacements, does not allow the agent to develop a deeper understanding of ingredient substitution. For this reason, we use different knowledge sources to augment the agent’s perception of the ingredients, using categorical and nutritional information. The ingredients are in practice annotated by qualitative labels that have a human intuitive form, i.e. “low fat”, “cattle diary”, “citrus fruit juice”, laying out the foundations that enable the agent to develop a human-centric understanding of the task. We will now go through the knowledge sources used.

We utilized the FoodOn ontology [1] to annotate the ingredients with **Food Product Categories**, that can be used for human-centric perception of the ingredients. FoodOn contains a very broad set of concepts allowing the identification and reference of food products and their constituents across the entire food supply chain. The biggest part of this ontology is not

expected to be useful for our task, while its size might make it very hard to put it to use altogether. We therefore used the same relevant subset of the ontology as the creators of FoodKG. This subset contains all child concepts of food products in a hierarchical form. Such concepts include for example “Montasio cheese” which is a child concept of “cow milk cheese”, which in turn is a child concept of “cattle dairy food product”. FoodKG already contains links between the ingredients of the Recipe1M dataset and FoodOn concepts. For more details regarding the FoodOn ontology, the subset used, or the entity linking between the ingredients and the concepts, please refer to the original work of [4].

The United States Department of Agriculture (USDA) periodically puts together catalogs of (commercial) food products or ingredients together with nutritional information. Such information describes 40 nutritional information dimensions including for example protein, cholesterol, sugar, etc. Although we do not currently use this information, our goal is to translate its quantitative form to human-centric categories describing **Nutritional Information**, like “low fat”, “zero sugar”, etc.

5.4 Agent Policies

As mentioned in Section 5.3, the ingredient substitution task requires intuitive humane understanding that can be expensive to either extract and formalize or to collect in large amounts. Instead, our approach is based on the premise that intuitive human-like understanding can be efficiently picked up by an AI system that interacts with a human. Our premise is based on prior work, suggesting that shared task-oriented understanding can be established via agent interaction [5]. We put forward agent policies, i.e. behavioral instructions of an agent, that will help it to i) learn from provided substitution examples and ii) have an active role on what to learn next, aiming both for effective and efficient learning. Specifically, the agent has access to a large set of substitution queries: a recipe and a particular ingredient that needs replacement. The agent does not know how to substitute the ingredient, but can ask the chef.

5.4.1 Learning Policy

We design the agent to learn based on the observations of its past interaction experiences. The interaction experiences consist of substitution examples provided by the chef. The agent records the frequencies of the observations and then uses them to probabilistically estimate the best ingredient substitutions for other scenarios. The assumption is that observations in past ingredient substitution examples are more probable to reoccur based on their frequency. Their frequency can then be used to estimate the probability of a plausible ingredient substitution.

The agent categorizes observations from ingredient substitution examples in three categories according to their assumed importance. A recorded observation relates two entities in an ordered manner. These entities can either be an ingredient or a property of an ingredient; each coming with a distinct IRI. The entities of recorded observations are always directed, relating an entity describing the original recipe to an entity describing the target substitute ingredient. The first and most important type of observation consists of only the ingredients directly involved in the substitution: the source ingredient and the target ingredient. We call this category of observations “**Ingredient to Ingredient**”. Figure 5.1 presents an example of such observation: “Rice Flour” → “Sorghum Flour”. The second category of observations involves every property combination of the two ingredients directly involved in the substitution. This category of observations is called “**Ingredient Property to Ingredient Property**”. An example of such observation present in Figure 5.1 is “Glutinous Rice Flour” → “Cereal Grain Food Product”. Last but not least, the third type of observations relates all property combinations of the recipe’s

ingredients (excluding the source ingredient) to all properties of the target ingredient. This last category of observations is called “**Recipe Property to Ingredient Property**”. An example of such observation is “Animal Food Product” $\rightarrow$ “Plant Seed Food Product”, presented in Figure 5.1.

The agent is able to estimate the likelihood of a candidate new ingredient, given a recipe and a source ingredient, by utilizing the recorded frequencies of observations. The different gravity of the three categories of observations is encapsulated with the use of score multipliers, during the calculation of the candidate’s substitution score. These multipliers are set to 100, 10, and 1, from first category of observations to last. Frequencies of observations are summed per category and are then multiplied with their corresponding score multiplier. The resulting products are added, providing the substitution score for a candidate target ingredient.

5.4.2 Active Learning Policy

Active learning policies require a query strategy, that is based on a utility function. Provided these two, the agent can choose the most informative substitution example to learn from next. Our query strategy falls under the category of representation-based query strategies, and specifically the density-based approaches [11]. Representation-based strategies focus on data samples, i.e. substitution examples, that reflect the largest part of the remaining training data possible. Additionally, the subcategory of density-based approaches select the data point that is “closer” to all other remaining data points in the training set.

The estimated informativeness of a substitution example is calculated using the utility function. Our querying strategy is exploitation-based, selecting substitution examples to query about, proportionally to their expected informativeness. In an attempt to combat the uncertainty of the estimated informativeness, we choose to apply an epsilon-greedy selection of the next sample, with an epsilon set to 0.1. This means, that 90% percent of the times, the next sample is selected proportionally to their estimated informativeness. In the remaining 10% of the cases, the next sample is selected randomly, based on a uniform distribution over all remaining training substitution examples.

Our utility function is calculated based on the popularity of source ingredients and their properties, among the remaining substitution examples in the training set. For example, in Figure 5.1, the source ingredient is “Rice Flour” and its properties are “Low Fat” and “Glutinous Rice Flour”. The frequency of this ingredient and these properties within the set of source ingredients in the training set, will affect the likelihood of selecting this substitution example to interact over next. Their popularity values are calculated by applying the logarithmic function on their frequency in that set. The logarithm is applied to the frequencies as a common technique for normalizing skewed data, and better reflect the informativeness of examples in comparable manners. The utility value of an example is calculated based on its source ingredients and its properties, as a sum. Specifically, we first sum the popularity of the properties and then add the popularity of the ingredient multiplied by 10, implying a difference of significance.

5.5 Experiments

In this section, we will present the current experiments of our ongoing study. First, we will describe or remind our experimental setup, together with the pairs of learning and active learning policies that our experiments compare. Then, we will present the results of the experiments together with their analysis.

5.5.1 Experimental Setup

As mentioned before, the agent has access to the recipes and source ingredients of the recipe, while the corresponding target ingredients are being revealed by the chef over time. We used the substitution samples from the validation set to estimate the performance of the agent policies for different parameters. The best parameters are reported in the description of our agent policies in the previous section, and here we report the agent performance on the test set, using those parameters. To reduce the effect of random chance in our evaluation, we repeat each experiment four times and report aggregated performances.

We utilized the Food product categories described in Section 5.3 as properties for the ingredients. Specifically, we used the most specific FoodOn food category of an ingredient as its property. This results to 16,780 ingredients being described by exactly one property each, from a property vocabulary of size 2,267. Additionally, we applied the assumption that a direct parent of a food category in FoodOn can also be used as an ingredient’s property. The outcome was a vocabulary of 4,081 properties and 37,466 ingredient-property relations.

As mentioned in Section 5.3, the ability of the agent to perform the task should be evaluated with respect to the amount of input needed from the domain expert. We therefore report the agent’s performance trajectory over the first 100 provided examples. The limit of 100 examples should reflect roughly the amount of substitution examples a chef can provide within a couple of hours, assuming just over 1 minute to read a recipe and come up with a substitution for a particular ingredient. We evaluate the agent’s task performance every 10 provided examples. The agent’s task performance is reflected using information retrieval metrics, i.e. $\text{hit}@k$, for $k \in \{1, 10, 100\}$ and Mean Reciprocal Rank (MRR). MRR is the harmonic mean of the retrieved ranks of the target ingredients over all samples of the test set. $\text{Hit}@k$ is the ratio of generated substitution suggestions where the target ingredient was found within the top k recommended substitutions. The efficiency of the agent policy is measured in terms of number of examples provided by the chef up to that point.

We additionally report the performance of the agent for the same combinations of policies, when provided with 1000 examples, to study their behavior in larger amounts of training data. Finally, in order to put the agent’s task performance into the appropriate perspective, we also present its performance when trained on the complete training set consisting of 38.142 examples. AL policy on the complete training set is not reported, since it only affects the agent’s training when the agent is exposed to part of the training set.

The applied substitution candidate ranking policies are based on recorder observations over ingredients or ingredient properties. Therefore, not the complete set of ingredients are ranked as candidates for every scenario, but only the ones that can be “justified” according to the agent’s recorded observations. The remaining, “unjustified”, ingredients are added at the bottom of the ranked recommendation list in a random order. We also report the average size of the justified substitution recommendation returned by an agent and the proportion of cases where the target ingredient was among them, signified by “**#Justified Recommendations**” and “**Target Justified**”, respectively.

The baseline learning policy is the one presented in [2] as “**LT+Freq**”. This baseline is the highest performing baseline in that study. Although their proposed GISMo model, performs better than LT+Freq, it is a data-hungry ML method and cannot be applied in our scenario. LT+Freq is based on the frequency of one ingredient substituting another in the provided training data, ignoring the ingredient properties. It then uses these frequencies to rank candidate ingredient substitutions. This would be the same as our learning policy, if we only used the “Ingredient to Ingredient” observations. We compare this baseline learning policy to our

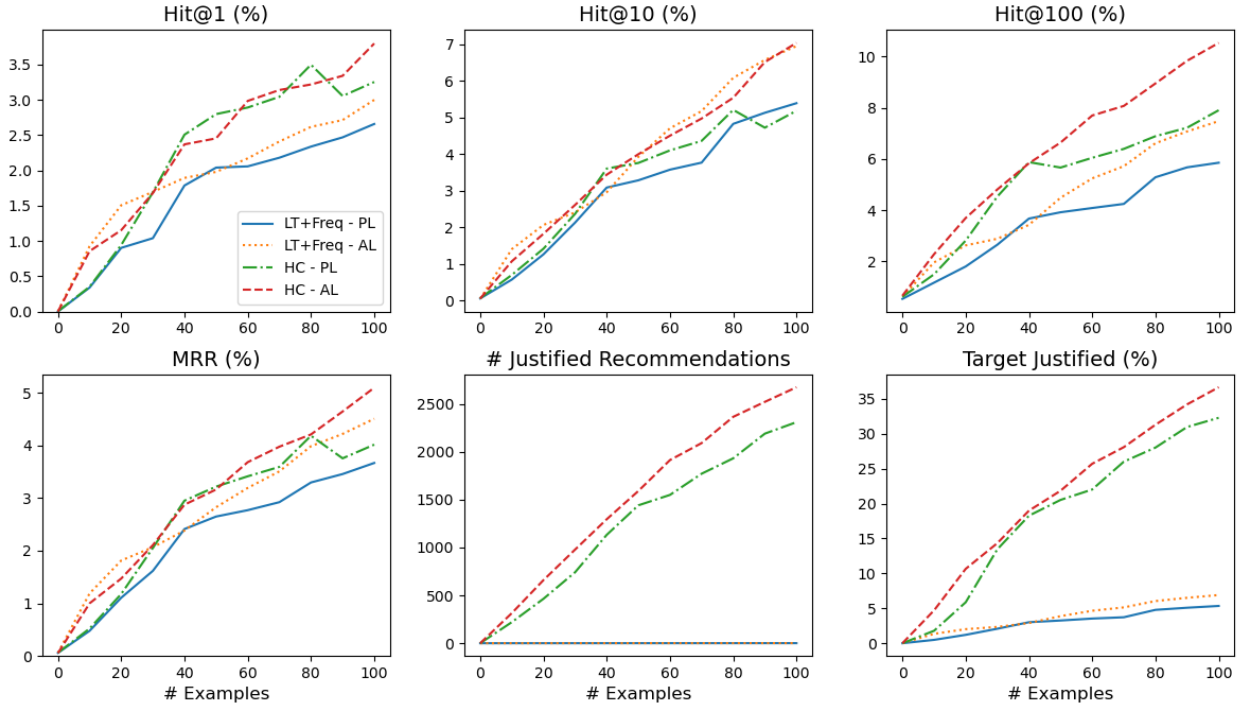


Figure 5.2: The agent performance on recommending ingredient substitutions, for different combinations of learning and AL policy combinations. Hit@k, for $k \in \{1, 10, 100\}$ and MRR exhibit task performance. Furthermore, “#Justified Recommendation” represents the number of ingredient recommendations that are based on prior experience observations of the agent, while “Target Justified” indicates the ratio of times the target ingredient was among these recommendations. It should be noted that the average number of justified recommendations, i.e. “#Justified Recommendations”, for the policy combinations of “LT+Freq - PL” and “LT+Freq - AL” reach a maximum of 0.8 and 1 after 100 examples, respectively.

suggested Human-centric learning policy, that additionally puts to use the properties of the ingredients, denoted as “**HC**”. Comparing these two methods should illustrate the benefits of using ingredient (Human-centric) properties, as it is performed in the current status of our study.

We compare our proposed active learning policy (AL) to a passive learning baseline (PL), where the substitution examples are provided in random order. With this comparison, we expect to see the added value of applying in active learning method in our experimental setting.

5.5.2 Experiment Results

In this section, we will go over the experimental findings that will allow us to draw conclusions over the presented agent policies for the task at hand. The results are presented separately for the efficiently learning scenario over 100 examples and the case where more training data are provided.

Figure 5.2 displays the agent’s performance for the efficient learning scenario, when only 100 examples are provided, for different combinations of learning and AL policies. Hit@1 shows small improvement of HC learning policies over LT+Freq. Specifically, this metric improves from 2.65% and 2.99% to 3.25% and 3.79% when using the HC learning policy, for passive and active learning, respectively. The hit@10 metric suggests that the two learning policies perform the same, but the AL policy can improve them from roughly 5.3% to 7%. Performance recorder on Hit@100 on the other hand, implied that both HC learning and AL can have an effect on the

Table 5.1: The performance of the different agent policy combinations after 1,000 examples and for the complete training set. Due to size limitations, please note that “# Just. Rec.” and “Target Just.” indicate the evaluation metrics “# Justified Recommendations” and “Target Justified”, respectively.

Learning	AL	# Ex.	Hit@1%	Hit@10%	Hit@100%	MRR%	# Just. Rec.	Target Just.%
LT+Freq	PL	1,000	9.78	22.66	24.94	13.71	6.40	24.47
LT+Freq	AL	1,000	10.15	23.26	26.02	14.52	6.60	25.54
HC	PL	1,000	8.80	13.20	17.96	10.69	8,065.2	78.70
HC	AL	1,000	8.45	15.10	20.33	11.13	7,847	77.82
LT+Freq	PL	Complete	19.59	50.99	63.80	29.79	64.1	64.33
HC	PL	Complete	10.99	17.37	21.43	13.59	13,771	97.08

performance, and can lead to almost doubling the performance when used together. notably, for this metric, “LT+Freq - PL” performs 5.85%, while “HC - AL” achieves 10.5% which is almost double. The agent’s capabilities, as captured by MRR, show small improvement either if we change LT+Freq to HC, or by choosing to use AL compared to PL. Finally, as expected, HC is able to relate more ingredients as candidate substitutions compared to LT+Freq, based on the broader range of observations recorder. Agents that learn using the HC policy are more probable to return the target ingredient among their justified recommendations, although this big difference is not captured to this extent on the task performance metrics. Concluding, on average both HC shows minor improvements over LT+Freq, while also the use of the proposed AL policy is beneficial, when the number of samples within the range of 100.

Table 5.1 presents the performance of the agent for the same combinations of policies for less efficient learning scenarios, when when provided with 1,000 examples or the complete training set of 38.142 examples. First, we should note that the AL method shows minor improvements when combined with the LT+Freq learning policy across all metrics, when 100 examples are provided. It does show small improvement when combined with the HC learning policy but only for the Hit@10 and Hit@100 evaluation metrics. Secondly, the baseline method LT+Freq exhibit superior performance over HC across all task performance metrics, while on average returning only 6.5 and 64 candidate ingredients compared to approximately 7.950 and 13.771, when trained on 1000 and 38.142 examples, respectively. Thirdly, it should be noted that the HC learning method manages to relate the target ingredient approximately 78% and 97% of the times when trained on 1,000 examples or the complete training set. Albeit LT+Freq manages to return the target ingredient in a justified manner in much fewer cases, i.e. roughly 25% and 64% of the times when exposed to 1,000 examples or the complete training set, it manages to achieve much higher performance. This allows us to say that LT+Freq provides much fewer suggestions, but is way more accurate, if trained on enough examples. Finally, as expected, AL has a smaller effect while the number of provided examples increases. This shows why it is important to use it when access to large number of training samples is difficult.

5.6 Conclusion and Future Work

To conclude, in this chapter we presented a new way of approaching the task of ingredient substitution, that is based on directly learning through interacting with an expert, aiming to mimic human-like understanding on this task. We presented how the interaction between an agent and a human can be done with clarity, by grounding the elementary aspects of the interactions. Furthermore, we presented ways to utilize knowledge sources, allowing the agent to have human-centric perception of the ingredients, and enabling it to develop human-centric understanding. Moreover, we put forward agent policies that can aim to learn both effectively and efficiently. Although the agent policies are still under development, we demonstrated that

they can improve existing baselines, even fractionally. Overall, our the proposed agent policies, i.e. HC and AL, help the agent perform better on the task when the number of provided examples is small, encouraging us to further continue this direction of study. On the other hand, when we provide 1000 examples or the complete training set to the agent, LT+Freq performs better than HC, and the use of the AL policy makes small difference. This suggests that only memorizing ingredient to ingredient observations can lead to better performance, allowing us to conclude that at least some ingredient properties affect negatively the agent’s recommendations.

Reminding that the presented work is still in progress, we aim to perform the following as next steps. First, all the substitution examples of the Recipe1MSubs benchmark should be grounded, i.e. represented using IRIs. This allows us to use the complete benchmark for better training and more accurate evaluation of our policies, while also establish the produced grounded version as a complete benchmark in the SW domain. In addition, we aim to incorporate more knowledge sources that can further extend the agent’s human-centric ingredient perception. Besides, it is our assumption that some ingredient properties are more important than others. It is therefore among our goals to apply methods that attempt to capture the difference in properties’ significance for our task, and to incorporate this knowledge in our learning and active learning policies. Other categories of learning or AL policies should also be tested for this task.

Chapter 5: References

- [BC1] Damion Dooley, Emma Griffiths, Gurinder Gosal, Pier Luigi Buttigieg, Robert Hoehndorf, Matthew Lange, Lynn Schriml, Fiona Brinkman, and William Hsiao. Foodon: a harmonized food ontology to increase global food traceability, quality control and data integration. *npj Science of Food*, 2, 12 2018.
- [BC2] Bahare Fatemi, Quentin Duval, Rohit Girdhar, Michal Drozdal, and Adriana Romero-Soriano. Learning to substitute ingredients in recipes. 2023.
- [BC3] Neelansh Garg, Apuroop Sethupathy, Rudraksh Tuwani, Rakhi Nk, Shubham Dokania, Arvind Iyer, Ayushi Gupta, Shubhra Agrawal, Navjot Singh, Shubham Shukla, Kriti Kathuria, Rahul Badhwar, Rakesh Kanji, Anupam Jain, Avneet Kaur, Rashmi Nagpal, and Ganesh Bagler. Flavordb: A database of flavor molecules. *Nucleic acids research*, 46, 10 2017.
- [BC4] Steven Haussmann, Oshani Wasana Seneviratne, Yu Chen, Yarden Ne’eman, James Codella, Ching-Hua Chen, Deborah L. McGuinness, and Mohammed J. Zaki. Foodkg: A semantics-driven knowledge graph for food recommendation. In *International Workshop on the Semantic Web*, 2019.
- [BC5] Nikolaos Kondylidis, Ilaria Tiddi, and Annette ten Teije. Establishing shared query understanding in an open multi-agent system. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS ’23, page 281–289, Richland, SC, 2023. International Foundation for Autonomous Agents and Multiagent Systems.
- [BC6] Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. Generating personalized recipes from historical user preferences. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5976–5982, Hong Kong, China, November 2019. Association for Computational Linguistics.
- [BC7] Javier Marín, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 07 2019.
- [BC8] Andrzej Nowak, Paul Lukowicz, and Pawel Horodecki. Assessing artificial intelligence for humanity: Will ai be the our biggest ever advance ? or the biggest threat [opinion]. *IEEE Technology and Society Magazine*, 37:26–34, 12 2018.
- [BC9] Donghyeon Park, Keonwoo Kim, Seoyoon Kim, Michael Spranger, and Jaewoo Kang. Flavorgraph: a large-scale food-chemical graph for generating food representations and recommending food pairings. *Scientific Reports*, 11, 01 2021.

- [BC10] Sola S. Shirai and HyeongSik Kim. Eat-pim: Substituting entities in procedural instructions using flow graphs and embeddings. In *The Semantic Web – ISWC 2022: 21st International Semantic Web Conference, Virtual Event, October 23–27, 2022, Proceedings*, page 161–178, Berlin, Heidelberg, 2022. Springer-Verlag.
- [BC11] Alaa Tharwat and Wolfram Schenck. A survey on active learning: State-of-the-art, practical challenges and research directions. *Mathematics*, 11(4), 2023.
- [BC12] Kirill Veselkov, Guadalupe Gonzalez, Shahad Aljifri, Dieter Galea, Reza Mirnezami, Jozef Youssef, Michael Bronstein, and Ivan Laponogov. Hyperfoods: Machine intelligent mapping of cancer-beating molecules in foods. *Scientific Reports*, 9, 07 2019.
- [BC13] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Part III

Semantic and Pragmatic Foundations

The symbolic approach to text understanding takes into account the extensive research that has been carried out by linguists specifically in the domains of cognitive semantics and pragmatics.

Cognitive semantics focuses on the conceptualisations that underlie a given language and how these conceptualisations are expressed using lexical and syntactic means. A typical example concerns the temporal framework (with tense, aspect, mood and modality) that languages assume and the verb forms that express choices for each of the perspectives in this framework. Another example concerns causal frames that describe the various dimensions in which an action causes other actions, or the argument structure which defines the roles of entities in the actions described by the verb. All of these issues play a prominent role in descriptions of everyday actions as found in recipes.

The paper by Anna Morbiato entitled 'The linguistic ingredients of recipes: aspect, causation, and argument structure' shows many concrete examples of the subtleties of cognitive semantics drawn from cooking recipes. The examples illustrate in detail the deep diversity across different languages, not only for the surface forms available for expressing conceptualisations but also the building blocks for these conceptualisations themselves.

Pragmatics is a branch of linguistics that studies how words and grammar guide the attention mechanisms of human readers and what should be considered foreground and background. Pragmatic indicators play a crucial role in understanding texts about everyday actions, such as recipes, in particular for tracking references from partial descriptions (as with pronouns, demonstratives or anaphora).

The paper by Anna Morbiato and Patrizia Cani entitled 'Pragmatics as the secret ingredient: a multilingual study of reference tracking and evolutive anaphors in culinary narratives.' introduces the pragmatic viewpoint, showing the crucial role it plays in anaphora resolution, especially when anaphors are of the type of zero or evolving anaphors. It shows how pragmatic devices operate through concrete examples drawn from recipes. It also shows that here also there is deep diversity across languages.

Chapter 6

The Linguistic Ingredients of Recipes: aspect, causation, and argument structure

The linguistic ingredients of recipes: aspect, causation, and argument structure

Abstract

Recipes outline goal-oriented procedures involving sequential actions with specific durations aimed at achieving precise outcomes. Ingredients are moved into different containers and undergo various processes, ultimately turning into new entities. The structure and complexity of recipes make them an exciting domain for AI and Natural Language Understanding research, for the challenges they pose and their potential for establishing benchmarks. Accurate understanding of procedures is paramount and involves correct interpretation of the sequence and duration of the various processes, as well as of the causal relationships expressed in each step. This paper aims to provide a thorough linguistic and textual analysis of recipe texts, exploring their nature, structure, and genre variations. It provides an in-depth examination of their primary linguistic functions, adopting a multi-language approach that involves analysing authentic recipes in typologically distinct languages such as English, Italian, and Chinese. The analysis consists of a function-form study of four mayor semantic frames, aimed at determining how different languages encode temporal and aspectual information, causal relationships such as caused motion and caused change of state, and resultant objects.

Keywords: recipes; procedural discourse; multi-language perspective; aspect; causation; caused motion; caused change of state; argument structure.

1. Introduction

Food is inherently human-centric. It plays a significant role in creating social relationships, shaping cultures, and providing occasions for social interaction. The topic of food description and preparation universally engages people all over the world, while interest in food writing has dramatically grown across the media. Recipes stand out as one of the primary and most characteristic formats of food writing. Within these texts, the necessary ingredients, quantities, and sequential processes are outlined to guide individuals in successfully cooking the dish. The language of recipes possesses unique and intrinsic features. These goal-oriented, procedural texts describe sets of actions carried out in a specific order over time to achieve a precise outcome. Ingredients undergo various transformations as they are moved into different containers and processed in countless ways, ultimately transforming into entirely different entities, such as a cream or a cake. The functional nature of such procedural texts is reflected in an array of linguistic functions which shape grammar, word order, and linguistic forms. Exploring the function-form relationships that constitute the texture of such texts from a cross-linguistic perspective is the core of this work.

The complexity and diversity of recipes make them an exciting domain for AI and Natural Language Understanding research and application, certainly so for the challenges they present. The first challenge relates to correctly interpreting the temporal and aspectual information connected to instructions: recipes are a series of sequential steps, where the order and timing of actions matter significantly. This provides a clear framework for testing an AI system's ability to understand and execute series of steps in a coherent and logical manner (thus presupposing correct extraction of temporal/aspectual frames). The second challenge is grasping the causal relationships described in the instructions themselves: what needs to be done and what changes need to happen in each step to reach the outcome (e.g., put some item in a specific location or act on some item to make it change its state or become a new entity). To do this, AI systems must comprehend and handle the causal dynamics processes between agents and items in each instruction (thus presupposing correct extraction causal frames). The third challenge is related to the endophoric and exophoric nature of coreferential devices. On one hand, recipes contain many references to contextual items (ingredients, tools) and presuppose a good awareness and mastering of world knowledge. On the other hand, coreference disambiguation of anaphoric means is not trivial, as they often consist of zero anaphors rather than overt pronouns or nouns: ingredients and 'resultant objects' are often unmentioned, as in '*Bake until crispy and golden.*' '*Pour until saturated.*' Hence, the text may be ambiguous and open to interpretation. Fourth, recipes often involve multiple modalities, including text (the recipe itself), images (ingredients and cooking process), and sometimes audio/video (voice instructions). Understanding information from these diverse sources require advanced AI techniques. Successful interpretation and integration of this multimodal information showcases the AI system's versatility and ability to handle diverse inputs. For these reasons, recipes are very useful to establish benchmarks for AI systems. When researchers and developers work on creating AI cooking assistants, they can use standardized recipe datasets and evaluation metrics to benchmark their system's performance against existing solutions or other research projects. However, a significant number of studies in the literature on cooking applications to AI are not based on solid linguistic analysis grounds.

This paper, together with the paper on anaphora (Morbiato, this volume), aims at addressing this research gap, in the hope to function as a linguistic basis for AI and NLU implementations. It aims to provide a thorough linguistic and textual analysis of recipe texts, exploring their nature, structure, genre variation, and primary linguistic functions, particularly those connected to the first and second of the challenges outlined above. The analysis consists of a function-form study of four mayor semantic frames, aimed at determining how different languages encode temporal and aspectual information (in terms of sequence and duration of actions and procedures), causal relationships, and result (caused motion, caused change of state, and creation of resultant objects), concluding with some remarks on argument structure and composition. It involves the examination of original recipes selected from international blog entries written languages that differ significantly from a typological perspective: an example of a Germanic language, i.e., English, a Romance language, i.e., Italian, and a Sinitic language, i.e., Chinese. This approach allows for the appreciation of both common traits that characterize recipes across languages and the identification of language-specific traits that may pose challenges for a multilingual AI architecture.

The rest of the paper is structured as follows. Section 2 provides an overview of the peculiarities of recipes as a text type. Section 3 presents the research objectives, data, and approach. Section 4 presents the function-form analysis of the temporal, aspectual, and causal frames as outlined above, along with some argument structural considerations. Lastly, section 5 draws some conclusions.

2. Recipes as a text type

As a well-known element of our daily life experience, recipes may appear relatively easy to define and describe. At a general level, recipe texts are written instructions that provide a step-by-step guide on how to prepare a specific dish or culinary creation. They are widely used in cookbooks, food blogs, cooking websites, and shows to share culinary knowledge and enable others to recreate the same dish. However, scholars have demonstrated that recipes are complex technical texts with unique linguistic features and a rich history. The method of writing a recipe has evolved over time, resulting in relatively uniform criteria only in recent times. Moreover, recipe texts display genre-related and cross-linguistic variation. What are the elements that make recipes a specific text type? This section is devoted to delving deeper into what a recipe is, its main characteristics in terms of structure, evolution, and genre-specific features.

2.1. Definition

According to the *Cambridge Online Dictionary*, a recipe is “a set of instructions telling you how to prepare and cook food, including a list of what food is needed for this”. The *Oxford Encyclopaedia of food and drinks in America* defines recipes as “the ideas and the instructions for handling foods and preparing particular dishes” (Waxman 2004, 347). As in dictionary definitions, several scholars who have defined the term “recipe” have focused their attention on the instructional and operational nature of such texts. Hayakawa ((1972, 58) states that recipes are “the best everyday example of operational definitions: if you follow a recipe you produce the food it defines”. Norrick (1983, 173) describes them as a set of goal-oriented rules in the class of immediately directive technical texts, also emphasizing their operational nature: the interest in studying recipes texts lies in their special purpose, namely reaching a “goal”, and in their language “not entirely within the boundaries of the general or core language”.

2.2. Evolution of the genre

Recipe texts have evolved over time, reflecting cultural practices, culinary traditions, and regional influences. Apparently, in ancient times recipes were mainly narrated in oral form. Hence, cookbooks were originally transcriptions of oral narratives, descriptions of cultural knowledge that might have been lost otherwise (Marrone 2015, 32). The earliest known written recipes date to 1730 BCE and were found in three cuneiform tablets in Mesopotamia (Byrd and Dunn 2020). Not surprisingly, one of the earliest traditions of cookbooks is that of Italy, with the *Liber de coquina*, dating back to the thirteenth century and written in Latin.

The need to describe and effectively explain foreign or exotic recipes seems connected with the evolution of the structure and codification of the text as we know it today. According to Sergio (2017, 196–97), the first cookbooks were quite concise and unspecific in their descriptions, as they were conceived for proficient readers, like cooks and servants, but starting from the 1300 they became more and more precise and procedural. The second half of the fifteenth century witnesses the emergence of cookbooks containing recipes presented in the form of procedural texts. In that period, a milestone cookbook appeared, which then set the standard for the genre. It was written by Cristoforo Messisbugo, who was paying service to the extravagant and powerful Renaissance court of the Estensi. The cookbook finely describes the luxurious banquets of the court and includes detailed recipes, featuring a big variety of specific lexical items, some measurements, and a procedural description of actions. As Sergio maintains, the necessity to include more details and specific instructions shows that the Estensi court, in their banquets, boasted a great variety of dishes, both traditional and “foreign”, that needed to be recorded and explained. Yet, before the eighteenth century, recipes were still considered more a reminder of ingredients for busy servants or proficient housewives than proper instructional texts. A seventeenth century Dutch cookbook

retains the informal, permissive features of a handwritten recipe. Syntactically, we can notice that the text above makes little use of features that are typical of recipes, such as null objects, locative, instrumental, and temporal clauses.

- (2) To make meatballs. Take veal with veal-fat chopped, add it to mace, nutmeg, salt, pepper, knead it together, then you can make meatballs from it as large or as small as you please also all of it is fried in the pan as one large meatball. Many take a few of the outside peels thinly pared of oranges or lemons. Cut very fine, it gives a very good smell and flavour. (Waxman 2004, 347)

Similarly, in America, the format and syntax of recipes changed significantly only after, as more and more settlers from foreign countries brought their culinary traditions with them to America, and household helpers started seeking job opportunities outside the family. Old East Asian cookbooks also display a lack of detailed information about time, location, instrument, and measurements in recipes. Yuan Mei's *Recipes from the Garden of Contentment*, a famous eighteenth century Chinese cookbook, presents a wide variety of recipes, described in a clearly sequential and procedural fashion, although still lacking bipartite structure and details.

The first European 'modern' cookbooks date back to the nineteenth century, when the genre finally became well established (Garzone 2017, 44). In the twentieth century, following mass literacy and the publication of popular cookbooks, more women started to find work outside the family and the interest for foreign food started to grow. Hence, details of food preparation stopped being self-apparent and a great deal of precision was required to prepare any dish (Waxman 2004, 348–49). A 1924 recipe exhibits almost the same syntactic structure and precision as current ones:

- (3) Corn Chowder

(list of ingredients) [...]

Fry onion in fat, add flour, stirring often, so that the onion may not burn; add two cups water and potatoes. Cook until the potatoes are soft; add corn and milk, and cook for five minutes. Season with salt and pepper, and serve. (Waxman 2004, 349)

Contemporary recipes are a result of social and historical processes that modified their original structure, making them true technical texts. The modern procedural nature of recipe texts, their syntactic structure, and richness of details lies in the necessity for the recipe to be understandable and precise, so that even an inexperienced or 'foreign' reader can obtain the desired result.

2.3. Textual organisation

A crucial aspect of contemporary recipes lies in their structured organization. Most of the structural and linguistic features of recipe texts relate to the procedural nature of such texts, which we present in section 2.3.1; section 2.3.2 outlines the main structural features of contemporary recipe texts, while section 2.3.3 explores some genre-related differences, which may be found, for example, in food blogs or web-related contents in general.

2.3.1. Recipes as procedural texts

Recipes are often mentioned as prototypical procedural discourse. Besides recipes, such text type includes a wide range of texts such as medical notices, itinerary guides, maintenance manuals, social behaviour recommendations, etc. (Fontan and Saint-Dizier 2008). Procedural discourse has a functional nature: to achieve a specific goal, one needs to follow specific instructions. Its importance lies in assisting users in learning and developing larger procedural knowledge and

providing a means for sharing knowledge with others distant in time and space (Hovde 2022). It is a goal-oriented type of discourse, as it consists of instructions that guide someone from a certain state of affairs to another. At a very abstract level, procedural discourse describes systems of states, and actions that change these systems creating new ones until the final goal is reached (Farkas 1999). Procedural texts are functional in their nature. Their usage implies the presence and interaction of three elements, the user, the text, and the object(s)/items(s) that might change in shape and state during the procedure. Thus, their particularity lies in this complex interplay of these three elements (Ganier 2012a).

Scholars have explored and defined the main rhetorical and logical features of streamlined step procedural texts. According to Farkas (1999), steps must be brief, containing one or two action statements and perhaps a short explanation or comment. A longer text might hinder comprehension as people's working memory is limited ('working memory constraint') (Ganier 2012b). Action statements are typically expressed using the imperative form, although a descriptive clause can precede or follow the main clause. In addition to short steps, procedures often utilize simple formatting, such as paragraphs, numbered steps, tables, or pictures. Preceding the steps, there is usually some introductory information, typically consisting of a title, a subtitle, and some sentences in the form of notes to introduce the procedure. The streamlined step procedure offers several essential advantages to the user, including enhanced legibility achieved through consistency of design and efficient cognitive processing due to the brevity and simplicity of instructions. This results in a high functional value of procedural texts and, ultimately, boosts efficiency in production (Farkas 1999). According to Longacre (1996), procedural discourse is characterized by a chronological linkage, rather than a logical linkage typical of other types of discourse, such as narrative discourse, and it lacks agent reference. Hence, in this type of discourse, the focus is on the action and its result, not on the agent. The order of the sentences is chronological and functional, with the purpose of displaying the best and most effective way to do something. Sentences are connected through a chain that strictly links the endpoint of one sentence to the beginning of the following sentence.

Scholars have also offered detailed analysis of the construction of streamlined step procedures in English, i.e., consisting of sets of mandatory or almost mandatory elements describing actions and states that contribute to the formation of the procedure. According to Farkas (1999), the components are: mandatory title, optional conceptual element describing or introducing the procedure, optional subtitle, procedural steps, and optional final notes where less important information is placed. The action statement, that is the main element of the procedural steps, can be employed on its own in a basic step or combined to different clauses, including locator clauses, instrumental clauses, conditional clauses, or purpose explanation clauses. Different kinds of clauses might be combined to create a complex step. Table 1 displays some features of procedural steps as theorized by Farkas.

Table 1: Features of steps of procedural discourse (Farkas 1999, 47)

Steps	Examples
Basic step	Click Format Painter.
Step with facilitating modifier (locator phrase)	<u>On</u> the File menu, click New.
Step with facilitating modifier (time phrase)	<u>When</u> the call is answered, lift the handset and begin talking.

Step with conditional clause and purpose Explanation clause	<u>If</u> your view of the tape is not expanded double. click the icon <u>to</u> display the tape volume.
---	---

Fontan and Saint Dizier (2008) define complex steps composed of clusters of instructions in English procedural texts as “instructional compounds”. These compounds are organized around a few main instructions, which are the instruction core, and enriched with several subordinate clauses that provide several layers of information, such as the goal and justification for the action, temporal information, causal and conditional relations etc. Table 2 contains some of the most recurrent elements in procedural texts as in Fontan and Saint Dizier. These elements, combined, create the complex structure of procedural texts.

Table 2: Instructional compound elements (Fontan and Saint-Dizier 2008, 118)

Elements	Examples
Goal and justification	<u>To change your mother card</u> , you must...
Instruction core	<u>Choose</u> specialized product for furniture.
Temporal structure	<u>Then</u> place it on a large clean surface Incorporate <u>and</u> stir.
Causal structure	Push the button <u>to</u> start the engine.

From the above, we can identify some major features of procedural texts:

- Clear structure and simple formatting
- Brief instructions
- Focus on the action and its goal/result
- Defocus on the agent
- Need to clearly express temporal sequence and aspect
- Need to clearly express causality

2.3.2. Structure

The conventional recipe text is characterised by a bipartite structure (Norrick 1983) comprising two key components: a comprehensive list of necessary ingredients and a detailed sequence of instructions for food preparation. Typically, recipes commence with a title indicating the dish and its main constituents, followed by an ingredient enumeration and a step-by-step guide to cooking. Additional segments like cooking tips, variations, or serving suggestions may also be included in certain recipes. This standardized layout facilitates easy navigation and effective transmission of culinary knowledge.

The ingredient list serves to clearly itemize all the prerequisites essential for the cooking process. Essentially, it functions as specialized instructions for gathering, imperative to the recipe's successful execution. These instructions ensure that all essential components are assembled and prepared prior to following the cooking steps. Consequently, the ingredient list is often considered a pivotal element in recipe texts, offering a concise and lucid overview of the necessary items required for the dish.

Step-by-step instructions primarily focus on the actions required to transform the ingredients into a finished product and how to utilize the necessary kitchen tools for the task, forming the 'instructional compound'. Within this instructions section, the steps are designed to reposition, transform, combine, or separate the ingredients, while also adjusting the temperature of the kitchen tools during the cooking process. Moreover, these steps are interlinked, with the output of one instruction serving as the input for the next. As a result, the successful execution of each instruction is crucial for achieving the desired outcome. Consistent with other procedural texts, instructions are typically presented in the form of imperative sentences that begin with a verb (Norrick 1983, 176). Cotter (1997) notes that the imperative establishes coherence among the discourse elements, thus providing a clear sequence of actions. This sequence is sometimes further reinforced with numbered sentences. When a non-imperative clause or sentence is included, it serves as a descriptive or evaluative marker. Descriptive clauses are often subordinate clauses, located at the beginning or end of an imperative clause and serve to supplement the text with additional information. These clauses are typically either locative, providing details about the location where the action takes place, instrumental, specifying the tools required to execute the action, or else providing variation tips. By incorporating these descriptive clauses into the recipe text, the author can enhance the reader's understanding of the recipe and provide additional context for the instructions.

Overall, the recipe format has been standardized and widely utilized across many different cultures and cuisines. This helps maintain clarity and consistency in culinary discourse, ensuring that readers can accurately reproduce the intended dish. However, some genre-related differences exist that will be presented in the next section.

2.3.3. Genre and register variation

Recipes are not confined to books. They have started to appear in other media, including television, magazines, cinema, and the internet (Floyd 2003 in Klenová (2010, 7)). Crucially, while professional cookbooks display a uniformity in grammatical and textual features, user-generated recipes exhibit greater variation in text structure, due to the lack of editorial control. They also show several linguistic features related to direct interaction between the blogger and the reader. While the main purpose of the cookbook is to provide simple and understandable cooking instructions, the food blogger also strives to create and gather an online community s/he can directly relate to. Hence, they contain a lot of elements that may pose difficulties for AI systems, such as descriptive parts, personal evaluations, or non strictly procedural information.

Kaneyasu and Kuhara (2020) conducted a comparative study of recipes in different media, showing that online user-generated contents differ greatly from edited and published recipes in cookbooks, so that they can be referred to as different sub-registers. Thus, they stress the importance of studying user-generated recipe texts on food blogs:

Before the internet era, most publicly available recipes were written by professionals (created by cooking experts, and written, revised, and edited by publishers) in the form of cookbooks. Since the late 1990s, more and more commercial recipes have been published on the internet; moreover, online recipes are increasingly produced by amateur writers. The trend justifies and calls for inclusion of online and user-generated varieties in a register study of recipes. (Kaneyasu and Kuhara 2020, 2)

In their study on English food blogs, Diemer and Frobenius (2013, 54–55) define them a “written asynchronous genre of computer mediated communication, that revolves around the preparation,

consumption and evaluation of food in all varieties and contexts”. As a subgenre of blogs, they present their characteristic design and technical features, like pictures, videos, and hyperlinks. These elements together with (often printable) recipes are embedded in a surrounding text, which narrates events related to food or food preparation, so food blogs often combine elements of printed cookbooks, diary-like texts, and web specific elements. For this reason, they have been defined as a “hybrid genre”. The scholars also notice that blog recipes often display frequent descriptive and evaluative elements that contribute to the informal character of the text, making it more personal and interactive. Modifying adjectives and adverbs like *some*, *about*, *little*, *large*, which make the recipe less precise, reduce its prescriptive strength, thus making the cooking more enjoyable, yet less machine language friendly. Evaluative statements marked by emotional terms like *good*, *fresh*, *favorite*, *pretty* create an emotional context, which although not necessary for the preparation of the dish, seem to facilitate emotive connection and interaction with the reader (2013, 64).

- (4) Here’s what I do. Take some beautiful Delicata squash, cut each one straight in half, and scoop out off the fuzzy bits in the middle and discard. (Diemer and Frobenius 2013, 64)

Wharton (2010) also focuses on the pragmatic aspects of recipes in cookbooks: through a study of speech acts, context, implicature, and inferential activity, he explores the ‘hidden meanings’ of this type of text and argues that a recipe is not just a procedural set of instructions, but a literary text that inspires the reader (2010: 73). Norrick (2011) also observes that recipes may have a mixed nature of storytelling narrative texts and procedural texts, with a ‘social’ purpose of establishing group membership and identity.

While recipes printed in contemporary cookbooks also contain some evaluative elements, food blogs make greater use of them. According to Lakoff (2006), compared to authors of recipes online, cookbooks writers tend to assume a less intimate relationship with the reader, positioning themselves as knowledgeable authoritative specialists and the reader as amateur. This difference in attitude and mood generally causes food blogs recipes to display a much more varied syntactic structure. In addition to the patterns commonly found in cookbooks recipes, food blogs feature a higher degree of structural variation, influenced mainly by elements such as audience address and imitation of spoken discourse, considered key features of the blog genre. Bloggers might construct minor sentences or insert full stops between elements of a sentence, thus resembling a casual spoken sentence. Other interaction features are the presence of discourse markers, hedges, and interjections (Diemer and Frobenius 2013, 69–71). Here is an example from our data:

- (5) (E.13-4) If you want it to be more like what you get out of the can, you can drain it in a colander set over a pot. I found that paper towels worked great for lining the colander.

According to Brdar-Szabó and Brdar (2009, 125), the use of different grammatical tenses to conceptualize instructions is due to a different degree of cooperation between the hearer and the writer. To create an intimate emotive bond with their reader, food bloggers might modify the classical imperative procedural form of the recipe and substitute it with a more colloquial, intimate text. Finally, in contrast to the cookbook authors, who are mostly “reserved”, food bloggers frequently refer to themselves, or ask questions to the reader, making large use of the personal pronouns *I* and *you*, instead of agentless imperative sentences, perhaps to make “viewers feel more included, or involved, and less as if they are being ordered to do something” (Wharton 2010, 68). Here is an example from the blog post we used for our research (somewhatsimple.com/pumpkin-roll-recipe). The first step is expressed by a non imperative instruction (introduced by the future form of the modal *want*). The second step begins with a descriptive clause which has non-procedural nature.

(6) How to Make Pumpkin Dinner Rolls

Prep – Preheat your oven to 375 degrees F. You'll also want to line your baking sheet with parchment paper or a [silicone baking mat](#).

Yeast Prep – I know a lot of people are intimidated by baking with yeast, but it's really simple once you get the hang of it! Start by dissolving the yeast in warm water and a pinch of sugar in your mixing bowl. Allow the yeast to sit for about 5 minutes until it's foamy.

With regard to Italian, Bosc (2017, 58) affirms that food blogs can be considered a “low-constraint discourse type text”, as the author feels free to express her/himself and provides the reader with possibilities of dialogue and comment. Therefore, the stiffness of the prescriptive text is lost in this communicative exchange between the writer and the reader, and new linguistic solutions are adopted. Bosc notices that typical textual features of Italian food blogs are clausal juxtaposition, cleft-sentences, and colloquial elements. She also notices the common use of personal pronouns and possessive adjectives. According to Bosc, these elements contribute to the informal and colloquial touch of food blog texts and reflect the “high-democraticity” of online language. Ultimately, Bosc affirms food blogs could be considered a “mixed” type of text, i.e., an intermediate form between written and oral language (Bosc 2017, 61–62).

These interactive, colloquial features are also present in East-Asian food blogs (Hsiao 2019): food bloggers use specific linguistic strategies to prompt and develop interaction with their readers. She points out three main linguistic strategies: narrative orientations, direct and indirect speech acts (like inviting promising, evaluating, and refuting), and direct reported speech. Narrative orientations present recipes as a piece of personal information, establishing an emotive bond with the reader that comments on the writer's story (Hsiao 2019: 498). Food bloggers make extensive use of personal pronouns to tell their stories and less use of subjectless imperative forms. Moreover, “instead of using imperatives, writers of popular food blogs tend to gravitate toward speech acts, to demonstrate professional knowledge and to show politeness toward the reader” (Hsiao 2019, 499).

To sum up, it seems that recipes in food blogs differ from cookbooks recipes in several ways: they contain more descriptive and evaluative elements, fewer subjectless imperative sentences, and more use of personal pronouns for self-reference, audience address, cooperation and solidarity, more questions and more elements resembling oral speech. It also seems that food blogs around the world display very similar linguistic features and an informal, colloquial tone.

In short, the structure of a recipe text may include the following elements:

- Title: The name of the dish, often placed at the top of the recipe to identify what will be prepared.
- Introduction: Short paragraph(s) that present the recipe from some perspectives, e.g., its history, evolution, or else some personal comments from the text author.
- Ingredients and quantities: A list of all the necessary ingredients required to make the dish along with the precise measurements or quantities of each ingredient needed for the recipe. Ingredients are typically listed in the order they will be used during the cooking process.
- Preparation Instructions: The step-by-step instructions detailing how to prepare and cook the dish. This section outlines each action to be taken, including chopping, mixing, baking, frying, and other cooking techniques.
- Cooking Times and Temperatures: Specific information on cooking times and temperatures, especially for baking or roasting recipes, to ensure the dish is cooked to perfection.
- Serving Suggestions: Optional suggestions on how to serve or garnish the finished dish for the best presentation and taste.
- Tips and Variations: Additional tips, tricks, and possible variations on the recipe to suit different tastes, dietary restrictions, or available ingredients.

- Nutritional Information: Some recipes also include nutritional facts, such as calorie count, macronutrient breakdown, and serving size, for those interested in dietary considerations.
- Further comments by the author.
- Pictures, videos, hyperlinks.

2.4. Linguistic functions

Comprehending recipes involves decoding different types of information, connected to specific linguistic functions. First, as recipes unfurl through a series of sequential steps, the precise order of actions (e.g., *wash and chop*) and the correct duration of the various processes (e.g., *bake for 30 minutes*) hold paramount importance. Thus, the expression of temporal sequence and (aspectual) duration are among the most important linguistic functions recipes present. Second, as ‘goal oriented’ technical-descriptive texts, recipes mainly describe the modification or combination of ingredients to create new items, until the outcome, i.e., the final dish, has been created. Ingredients are transformed through direct actions carried out by the subject, like cutting, rolling, or kneading, or through ‘indirect’ processes like raising, cooling, or baking, which modify their shape, taste, and other inner properties. Hence, the expression of causality, in terms of caused change of state (e.g., *refrigerate*), as well as of result, i.e., the creation of a new object (e.g., *cut into pieces*), is to be considered the most important function recipe texts have always had. Other important information includes details about the location where ingredients should be placed or where the action should be carried out, in other words, the expression of caused motion and resultant location (e.g., *put in the oven*). Indeed, locative elements expressing this linguistic function represent one of the most important features of contemporary recipes.

3. The study: objectives, corpus, and framework

This study aims to explore how different languages encode the linguistic functions identified in the previous section. The first objective is to identify and account for linguistic forms that encode temporal and aspectual frames of sequentiality and of duration. The second objective is to correctly describe and extract causal frames: on one hand, caused motion, on the other, caused change of state or creation of new objects. Given the inherent challenges displayed by food blog recipes, as described above, the corpus of this study is composed of food blogs entries. The dataset comprises recipes sourced from international food blogs (somewhatsimple.com, pandipane.blogspot.com, and meishichina.com) detailing the process of crafting pumpkin bread rolls. The wide range of online texts available allowed us to discover nearly identical original recipes in different languages. As non-edited, user-generated recipes, they feature both blog-like conversational features and traditional cookbooks-like procedural features.

The analysis adopts Construction Grammar (Goldberg 1995, 2006, Croft 2001, Kay & Fillmore 1999, among others) as its main framework, a well-established approach to language and linguistic analysis, with a representative body of literature that validates the theory’s basic tenets and its successful application to account for wide range of phenomena. CxG holds that different types of linguistic units are instances of constructions, including phrases, basic sentences, and idioms. Constructions are defined as pairings of form and function and recognised as independent theoretical entities, as they carry meaning independently of the lexical elements they occur with. For the same meaning, there is a diverse range of constructions in different languages. This framework also stresses the compositional and creative components of language: when verbs are used in novel contexts or unfamiliar ways, like the verb “to cook” in (7) it is the sentence structure, and not only the lexical compositional meaning of the verb, that provides the reader with a link between a form and its meaning.

- (7) (a) The chicken cooked all night. (Intransitive inchoative)
 (b) Pat cooked the steaks. (Transitive)
 (c) Pat cooked the steaks well-done. (Resultative)
 (d) Pat cooks. (Deprofiled object)
 (e) Pat cooked Chris some dinner. (Ditransitive)
 (f) Pat cooked her way into the Illinois State bake-off. (Way construction)

While the meaning of “cook” is the same – a transitive, bivalent verb that subcategorizes for an agent and a patient, conveying the concept of preparation of food through the application of heat – each sentence above conveys a different meaning, including caused change of state, caused motion, or transfer, uniquely through the skeletal constructional pattern (Goldberg 1995, 10). This specific aspect is of particular significance for our analysis and will be explored in Section 4.4. Moreover, this framework allows examining the range of linguistic means used to express a specific function in different languages, as well as conducting cross-linguistic comparison of typologically and structurally different languages, such as those targeted in this study. Most importantly, constructions can be implemented in computational formalisms, and in particular Fluid Construction Grammar (FCG) (Steels 2011; van Trijp, Beuls, and Van Eecke 2022), thus making this study suitable for application in computational linguistics, NLU, and AI.

4. The analysis: form-meaning pairs

In what follows, form-meaning pairs connected to the three functions identified above will be described and discussed, and specifically: temporal and aspectual frames of temporal sequence and duration, and causal frames of caused motion and of caused change of state. Finally, some argument structural considerations will be made that highlight the compositional nature of the linguistic forms associated with these functions.

4.1. Temporal and aspectual frames

Extracting temporal and aspectual frames is essential for the correct execution of the recipe. In recipes, two types of temporal/aspectual information can be identified. The first determines the sequence in which actions need to be carried out ($A1 > A2 > A3$); the second, the duration of a specific action ($A1$ for X time).

Sequences of actions are most times reflected in the order of instructional clauses, thereby displaying the kind of time sequence diagrammatic iconicity Haiman (1980) discusses. As said earlier, procedural discourse is characterised by a chronological order, and not by a logical linkage. The order of the sentences is functional, its purpose being to display the best, most effective way to do something. Some recipes also display numbered instructional sentences, which also helps identifying the correct order. However, some cross-linguistic differences can be identified. In Italian and English, each action is usually encoded in a different clause or sentence; two actions may also be conjoined through the coordinating conjunction *and* or reinforced through adverbs of time.

- (8) (E.29) Let the pumpkin cool **and then** scoop the meat out into a blender.
- (9) (I.10) tagli-ar-la a grosse fette **e** cuoc-er-le in una pentola
 cut-INF-3F.SG in big slices and cook-INF-3F.PL in a pan
 ‘cut it in large slices and cook them in pan’

In Chinese, conversely, lists of VPs describing sets of actions might appear in the same clause or sentence, while the use of coordinating conjunctions to express sequentiality is not required. In fact, whenever inferable from the context, no clause connectives are used. This is mainly due to the presence of discourse-structural factors, like the use of discourse topics or topic chains (see Morbiato, this volume).

(10) (C.9) (a) 南瓜泥，用南瓜去皮切块蒸熟，

nánguā ní yòng nánguā qù pí qiē kuài zhēng-shú
pumpkin puree TOP use pumpkin remove skin cut piece steam-cook
'As for the pumpkin puree, peel the pumpkin, cut it in pieces and steam it'

Coordination conjunctions and time adverbials are also cues that provide essential temporal and aspectual information to successfully carry out the recipe:

(11) (E.5) **When** they're all clean, put them face down on a baking sheet

(12) (E.43) **Once** the dough comes together and is slightly sticky to the touch, you'll know it's ready.

(13) (I.16) **Quando sar-à** incorda-t-o aggiung-ete a filo l' olio
when be.FUT.3SG blend-PTCP-M.3SG add-IMP.3PL at thread the oil
'When the dough is blended, slowly add some oil'

In English and Italian, time adverbials or conjunctions like *then*, *once*, *when*, and *until* are often used in combination with verbal tense and mood inflection in the form of *consecutio temporum* as temporal/aspectual frames that express sequentiality (Chinese, conversely, presents no verbal morphology). Verbs conjugated in the indicative future simple, present perfect or past participle contribute conveying a sense of order in the clauses. In (I.19) (a), sequentiality is conveyed though the past participle in the first subordinate.

(14) (I.19) (a) Tol-to da-l frigo fate torn-are a temperatura ambiente per un'ora
remove-PTCP from-the fridge let return-INF at temperature room for 1 hour
Lit. 'Removed from the fridge, let (the dough) warm up to room temperature for one hour'

Crucially, verbal morphology encoding *consecutio temporum* can override the iconic sequence of actions, e.g., with prepositions like *after* in the example below: here, the second predicate denotes the first action to be carried out and vice-versa:

(15) [...] let your dough rest after mixing for ten to thirty minutes. (Source: thebakersguide.com)

The second aspectual frame expresses how long a specific action needs to be carried out, which is also a crucial function in recipes. Duration can be expressed through different forms, such as a time phrase (usually a PP introduced by *for/per*) or an expression of resultant state (usually a PP or a subordinate introduced by *until/fin*o a che, *when/quando*, or *once/una volta che*; Chinese tends to use 至 *zhì* 'until').

(16) (E.37) Allow the yeast to sit **for** about 5 minutes **until** it's foamy.

(17) (E.46) [...] let it rise **for** an hour.

(18) (E.28) They are finished **when** a fork slides in easily

(19) (I.16) impast-ate **fino.a.che** l' impasto non sarà omogeneo
knead-IMP.PL until the dough NEG be.FUT.1SG homogeneous
'Knead until the dough becomes homogeneous, smooth and elastic.'

- (20) (C.19) 发酵至两倍大
 fājiào zhì liǎng bèi dà
 ferment until two time big
 ‘let it rise until it doubles in volume’

In fact, such frames can encode both sequentiality and duration, as in the example below:

- (21) (E.42) **Once** that has mixed together, add 1/4 cup of additional flour at a time.

4.2. Causation frame 1: the caused-motion construction

Change of location may be self-instantiated (as in *I enter the room*) or externally caused (as in *I put the cake in the oven*). The latter sentence is an example of the caused motion construction, which encodes the idea that an agent causes a patient argument (as e.g., a recipe ingredient) to move along a path and reach a goal, as in *Put the ice into the blender*. It displays the following semantic representation:

Agent (X) causes Patient (Y) to move to Goal (Z)

The semantic representation captures the meaning of the construction, which in different languages may map onto different syntactic structures. Goldberg (1995) captures this with the scheme in Figure 1, which includes both the semantic and the syntactic structure, along with the mapping between the two.

Sem	CAUSE-MOVE	<	cause	path	patient >
	↓	↓	↓	↓	
Syn	V	SUBJ(X)	OBL _{pp} (Z)	OBJ(Y)	

Figure 1: Caused motion construction in English, Goldberg (1995)

This schema allows to compare linearization patterns easily and effectively across languages: in English, for example, the caused motion construction often realized as follows:

- (22) (E.49) place them on your prepared baking sheet.
 V(place) + Obj(patient) + PP(goal)

In the caused motion construction, the verb must be non-stative and the oblique must be a directional phrase. Sentences encoding resultant location display verbs that lexically encode motion, like *put* or *place*, an object (the patient), and prepositional phrase encoding the resultant location (which is the most salient, focal part of the utterance).

- (23) (E.46) When smooth, place the dough into a greased bowl and let it rise for an hour.

Besides verbs lexically encoding motion, change of state verbs, such as *scoop*, *drain*, and *dissolve*, are also used.

- (24) (E.8) [...] and then scoop the meat out into a blender.
 (25) (E.13) [...] you can drain it in a colander set over a pot.
 (26) (E.36) Start by dissolving the yeast in warm water and a pinch of sugar in your mixing bowl.

Crucially, neither these verbs nor the locative phrases inherently encode caused-motion semantics, yet when combined in the construction, the pattern is understood as caused motion. We will discuss this in more detail in Section 4.4.

In Italian, the caused motion construction is encoded in a manner very similar to English. The most common pattern involves transitive motion verbs of putting, like *mettere* or *porre* ‘put’ plus PPs, as in the examples below. Nonetheless, while English displays a rigid constituent order, Italian features more possible patterns: the resultant location can be placed either before (I.14) or after the object (I.10). The object (patient) may also be left unmentioned (I.21) (this also happens in English).

(27) (I.10) (c) o metter-le in forno [...]

or put-them in oven

‘or put them in the oven[...],’

(28) (I.14) (a) Mett-ete in una ciotola il quantitativo di polpa di zucca (120gr)

put-IMP-PL in a bowl the quantity of pulp of pumpkin 120 grams

‘Put in the bowl the pumpkin puree (120 gr)’

(29) (I.21) Pon-ete a lievitare direttamente su-lla placca da forno.

put-IMP.PL to rise directly on-the tray of oven

‘Put the dough to rise on the oven tray wrapped with parchment paper’

This can be accounted for with the end-weight principle: the longer the constituent, the more towards the end it is located (Levin and Rappaport Hovav 2005). In (I.10), the object is a clitic pronoun and is placed before the PP denoting the resultant location. Conversely, longer objects, like *il quantitativo di polpa di zucca (120gr)* in (I.14), occur in the sentence-final position. Information-structural considerations are also at play: in both examples, the last constituent expresses the newest, most salient information. Similarly, in (I.21), the patient is omitted because inferable from the previous context.

As in English, some of the sentences encoding resultant location do not employ verbs lexically coding motion, but rather caused change of state verbs like *cook* in (I.10).

(30) (I.10) (b)[...] cuoc-er-le in una pentola con un dito d' acqua

cook-INF-3F.PL in a pan with a finger of water

‘[...]cook them in a pan with a little water’

In the Italian recipe we can also notice the appearance of clauses that code a change in location but do not mention the resultant location.

(31) (I.19) (a) Tol-to da-l frigo fate torn-are a temperatura ambiente per un'ora

remove-PTCP from-the fridge let return-INF at temperature room for 1 hour

‘Remove the dough from the fridge and let it warm up to room temperature for one hour’ s

In (I.19) (a), for example, in the clause *tolto dal frigo*, the verb *remove* is conjugated in the past participle and followed by a PP coding movement from a location. Although it clearly expresses a change in location, this clause leaves the resultant location unmentioned, since the author does not need to specify a particular place, which can be easily inferred through contextual knowledge.

Differently to English and Italian, the word order of the Chinese caused-motion construction varies a lot depending on information structural factors. In (C.10), the patient is preverbal, because highly active in the reader's mind (in the recipe, the list of ingredients is right above). In (C.20), it

is marked by Chinese a patient marking morpheme (like *bǎ* 把 or *jiāng* 将), that introduce the patient preverbally. In many cases, the patient is left unmentioned, in that it encodes given information, as in (C.9). In other cases, it is the resultant location that is left implicit (C.12), again because it is contextually or co-textually inferable:

(32) (C.10) 然后除黄油外所有材料放入厨师机内。

ránhòu chú huángyóu wài suǒyǒu cáiliào fàng-rù chúshījī-nèi
 then except butter out all ingredient put-insert mixer-in
 ‘Then put all of the ingredients in the mixer except the butter.’

(33) (C.20)(a) 冬天需要将碗座入温水上面

dōngtiān xūyào jiāng wǎn zuò-rù wēn shuǐ shàngmiàn
 winter need PM bowl sit-insert warm water above
 ‘In the winter, you’ll need to put the dough bowl into a bowl of warm water’

(34) (C.9)(b) 然后入榨汁机

ránhòu rù zhàzhījī
 then insert juicer
 ‘then put it in the juicer’

(35) (C.12) 加入黄油[...]

jiā-rù huángyóu[...]
 add-insert butter
 ‘Insert the butter[...]

In terms of predicates, we can notice that most consist of V_1V_2 compounds, such as *fàng-rù* ‘put-insert’ (C.10), *zuò-rù* in (C.20) or *jiā-rù* ‘add-insert’ in (C.12), where V_1 encodes the manner of motion/action, while V_2 is the motion verb encoding the direction and subcategorizes for a goal argument without the need of a preposition.

Table 3 summarizes the form-meaning pairs identified so far for the caused motion construction in the surveyed languages.

Table 3: Caused motion construction

Semantic representation (meaning):	Cause Patient (Y) to move to Goal (Z)
Syntactic realizations (forms):	
English	V + Obj(patient) + PP(goal) V + PP(goal)
Italian	V + Obj(patient) + PP(goal) V + PP(goal) + Obj(patient) V + PP(goal)
Chinese	NP(patient) + V(put) + PP(goal) Patient marker + NP(patient) + V(put) + PP(goal) V(put) + NP(goal) V(put) + NP(patient)

4.3. Causation frame 2: the resultative construction

Recipe texts generally feature lists of subject-less sentences that include an object undergoing a transformation, the action that transforms it, and the object resulting from this action, such as (36). This transformation might affect the object's shape, size, or inner properties.

(36) Cut the carrots into small dices.

According to Goldberg (1995), the sentence above is an instance of the *resultative construction*. It involves a caused change of state underwent by a patient argument, for example an ingredient, as a result of the action denoted by the verb. Sometimes this may entail the creation of a new item, i.e., a resultant object, like *small dices* above. Here, a *resultant object* is intended as the result of any transformation occurring to the ingredients following either a direct action carried out by the agent on the patient, or a gradual process derived by letting the ingredient in a certain state until the necessary change has occurred. Figure 2 is the semantic-syntactic structure, or the correlation between form and meaning, of the resultative construction, according to Goldberg (1995).

Sem	CAUSE-BECOME	{	agent	patient	result}
	↓		↓	↓	↓
Syn	V		SUBJ(X)	OBJ(Y)	OBL/RP(Z)

Figure 2: Resultative Construction in English, Goldberg (1995)

In our English data, the resultant object of is usually expressed through an oblique introduced by the prepositions *in* or *into*. The pattern is noted as follows: [V + Obj + RP (NP)].

(37) (E.3) Cut the pumpkins in half, top to bottom.

(38) (E.47) [...]then you'll divide it into two pieces.

(39) (E.48) Roll each piece out into a circle and cut each circle into twelve wedges.

(40) (E.49) Roll each wedge up into a crescent shape and place them on your prepared baking sheet.

The resultative phrase (RP) is expressed through a PP that is generally introduced by *in* or *into*, which conveys a metaphorical idea of change of location through change of state. Goldberg (1995, 83) states that the resultative construction can be semantically related to the caused-motion construction, since the change of state it codifies is intended as a metaphorical change of location (as in *The jello went from liquid to solid in a matter of minutes*).

In Italian, the structures encoding resultant objects are similar, but display some differences on the syntactic and information structural level.

(41) (I.10)(b) tagliar-la a grosse fette
cut-it in big slices
‘cut it in large slices’
Cause (Y patient) to become (Z Result-goal)
V(cut) + Obj + RP (NP)

(42) (I.19)(d) taglia-te i triangoli che andrete ad arrotolare per fare le brioche
cut-IMP.PL thetriangles that go.FUT.PL to roll.up to make the brioches
‘and cut out the triangles that you’re going to roll up to make the brioches’

VP1	VP2
Cause Y1 (Patient) to become Z (Result-goal)	(Y Patient) that (X agent) causes to become
(Z Result-goal)	
V(cut) + RP (NP)	Obj(patient) + V(roll up) + RP(NP)

The RP can be a PP introduced by the prepositions *a* or *in* (I.10), or else an NP as in (I.19) (d). Specifically, in (I.19) (d) the resultative object, i.e., ‘the triangles’, is placed in the patient slot instead of the real patient, i.e., ‘the dough’. Therefore, instead of writing ‘Cut the dough into triangles’, the author writes ‘cut the triangles’. This VP is then connected through a relative phrase to another VP featuring the verb ‘roll up’ and another resultant object, ‘the brioches’ introduced by a PP. Here, a single sentence expresses two steps through two VPs, whereby the resultant object of the first step becomes the patient of the second through a relative subordinate clause. This sentence, is perfectly understandable since the reader can infer that the triangles have to be cut out of the dough, even if the latter it is not mentioned. The text also features a hyperlink that helps the reader gather more information about the procedure. Other possible patterns involve the presence of infinitive subordinate clauses, either with infinite or gerund verb forms:

- (43) (I.17) (a) Pirl-ate un po' l' impasto a form-are una bella palla liscia.
spin.IMP.PL a bit thedough to form-INF a good ball smooth
‘Spin the dough a little bit to form a nice, smooth ball.’
- (44) (I.14)(c) ora mescol-ate bene il tutto ottenendo un composto cremoso.
now mix-IMP.PL well the everything obtaining a mixture creamy
‘now mix everything very well to obtain a creamy mixture’

In Chinese, the element representing the informational focus, i.e. the new, most salient part of the message, are placed sentence-finally. In recipes, the resultant object and the resultant location are the focal elements, hence often appear in post-verbal position. The patient, on the other hand, in informationally known and active, and is often located pre-verbally or left unmentioned (cf. Morbiato, this volume).

- (45) 把南瓜切成块
bǎ nánguā qiè-chéng kuài
PM pumpkin cut-become piece
‘cut the pumpkin into pieces’

- (46) (C.21)(b) Ø 揉成圆形
róu-chéng yuán xíng
knead-become round shape
‘knead them into a round shape’

Table 4 summarizes the form-meaning pairs identified so far for the resultative construction in the surveyed languages.

Table 4: Resultative construction - Cause (Y Patient) to become (Z Result-goal)

Semantic representation (meaning):	Cause Patient (Y) to move to Goal (Z)
Syntactic realizations (forms):	
English	V + Obj(patient) + PP(result/goal)
Italian	V + Obj(patient) + PP(result /goal)

	V + Obj(patient) + subordinate (V + Obj(result /goal)) V + Obj(result /goal)
Chinese	NP(patient) + VP(_v manner+ _v result) + NP(result /goal) Patient marker + NP NP(patient) + VP(_v manner+ _v result) + NP(result /goal) VP(_v manner+ _v result) + NP(result /goal)

4.4. Argument structure and coercion

As seen in in both causative frames analysed, verbs that enter these constructions do not always subcategorize for all the linguistic elements that compose the construction. For example, change of state verbs, like *scoop*, *slice* or *grate*, are transitive verbs of actions on things (verbs of creation and transformation, or verbs of cutting, separating, and combining) (Levin 1993). They all display following argument structure: <agent, patient> and do not inherently subcategorize for a goal/resultant location, unlike verbs of putting (Levin 1993). Yet, they also appear in a caused-motion construction.

- (47) (E.8)[...] and then scoop the meat out into a blender.
Cause Patient (Y) to move to Goal (Z)

In this case, the caused motion meaning is provided by the construction and not by the verb on its own. Goldberg exemplifies this with examples like the following (Goldberg 1995, 171):

- (48) The butcher sliced the salami onto the wax paper.
(49) Joey grated the cheese onto a serving plate.

As Goldberg and other scholars claim, this can be accounted in terms of coercion. Certainly, world knowledge and inference are at play: the semantics of the verbs implies a sense of predictable incidental motion. The verb *to slice* is a verb of cutting, while the verb *to grate* is a verb of separating, more specifically a split verb (Levin 1993, 166): the actions of carving and grating semantically implies that the carved/grated part moves away from its original location into a new location. Therefore, the locative phrase is allowed because it specifies the path of this incidental motion. Here are further examples from our data:

- (50) (E.39) Next, you'll want to add the remaining brown sugar, salt, egg yolk, egg, pumpkin puree, pumpkin pie spice, and a quarter cup of butter to the bowl.
(51) (E.13) you can drain it in a colander set.
(52) (E.69) In a large bowl or the bowl of a stand mixer, dissolve the yeast in the warm water with a pinch of brown sugar.

Examples above show that different types of non-motion verbs may enter this construction, including

Specifically, the verb *drain*, which doesn't code cause-motion independently, in this construction acquires a perfectly understandable directional meaning, as general knowledge suggests that, when this action is carried out, part of the object is detached from its previous location. As we can observe below, the sentence could work perfectly even without a directional phrase, but the author prefers adding locative details for clarity and convenience.

(53) [...] you can drain it.

The caused-motion construction, therefore, provides a change of state action the possibility to become a directional action. It is also worth noticing that sentences containing the verb *add* or *stir in* do not always contain DPs.

(54) (E.41) Slowly Add Flour – You will only want to stir in 2 cups of flour to start.

For example (E.39) specifies the resultant location, *the bowl*, but (E.41) doesn't mention it, since probably the author feels the location is easily inferable from the context. Although not specifying endpoint location, these sentences might still be considered directional, as they code the movement of an object along a path to an endpoint location. The caused-motion interpretation is given in this case by the pragmatic situational knowledge of the act as incidental motion of the object, directly and intentionally caused by the agent.

Crucially, this construction allows intransitive verbs, as well, i.e., verbs that only subcategorize for an agent, as e.g., 'to laugh' in *They laughed the poor guy out of the room*: here, *the poor guy* is understood to move out of the room because of the shame produced by the laughing. Since this interpretation cannot be attributed to any of the elements of the expression and cannot be derived compositionally, the caused-motion interpretation is attributed to the caused motion construction. Goldberg argues this is an example of coercion. The location encoded by the locative phrase acquires a meaning of endpoint focus, i.e., the endpoint of the path designated by the locative phrase. The verbs, on the other hand, although not lexically coding motion, acquire a motion interpretation (Goldberg 1995, 159, 175).

Similar considerations hold for Italian, although only one example was found in our data, with the verb of transformation *cuocere* 'to cook':

- (55) (I.10) (a) tagli-ar-la a grosse fette e cuoc-er-le in una pentola con un dito d'acqua
cut-INF-3SG in big slices and cook- INF-3PL in a pan with a finger of water
(56) (I.15)(a) Aggiungete la farina e gradualmente inserite la restante acqua.
add the flour and gradually insert the remaining water
'Add the flour and gradually insert the remaining water'

Verbs of transformation, like *cook* and *fry*, imply that the change of state must happen using some tool which inherently encodes a location (like a pan). Again, both the construction and pragmatic situational knowledge are at play: the author does not feel the need to specify that the patient needs to be placed inside the instrument before the process can start, as the construction as well as world knowledge provide this added meaning.

5. Concluding remarks

This article surveyed the characteristics, structure, functions, and variations found within recipe texts. These texts serve as procedural, goal-oriented descriptions that outline a sequence of actions occurring within a specific timeframe, at a particular location, and resulting in distinct outcomes. These functions encompass the encoding temporal sequences and durations of actions, as well as that of caused motion towards resultant locations and caused changes of state (which may lead to the creation of resultant objects). Regarding the representation of temporal action sequences, it is typically encoded through a list of agentless imperative sentences, often introduced by time adverbials. However, in English and Italian, each sentence usually conveys a single action, whereas in the Chinese recipe, a series of actions might be listed within the same sentence without the use of coordinating conjunctions or punctuation. Furthermore, both English and Italian recipes utilize

additional tools such as verbal agreement (*consecutio temporum*) and clause coordination to express sequentiality. In contrast, Chinese recipes predominantly rely on time adverbials and listing to convey sequence. Time duration is commonly expressed in all three languages through prepositional phrases (PPs).

The expression of caused motion towards resultant locations is achieved through a specific construction known as the caused-motion construction. The analysis suggests that the three languages generally employ distinct forms and phrasal patterns to express such construction. In English, this construction typically follows the word order [Sbj + V + Obj + DP], often utilizing a directional preposition followed by a noun phrase (NP) to indicate the resultant location. Meanwhile, Italian exhibits a variety of patterns. The choice of the pattern is subject to information-structural or end weight factors. In Chinese, the patient/object is placed preverbally, followed by directional verb compounds comprising an action verb and a directional verb to convey resultant location.

Caused changes of state and the creation of resultant objects are expressed through the resultative construction, that describes the transformation of a patient argument due to the action denoted by the verb(s). In English, the primary pattern is [V + Obj + RP(NP)]. While English and Italian recipes share similar approaches to expressing resultant objects, Italian may occasionally use alternative patterns. In Chinese, resultative patterns feature a different word order: the patient is positioned preverbally, followed by a resultative compound where the first verb conveys the action and the second introduces the resultant object as an NP.

Notes: For academic purposes, Anna Morbiato is responsible for Sections 1, 2, 2.2, 2.3, 2.3.2, 2.4, 3, 4, 4.1, 4.4, and 5; Patrizia Cani is responsible for Sections 2.1, 2.2, 2.3.1, 2.3.3, 4.2 and 4.3. Part of this research was drawn from Patrizia Cani's MA thesis (2022). In this paper, we use the term 'Chinese' to refer to Pǔtōnghuà, the standard language of the PRC. Throughout the paper, glosses follow the general guidelines of the Leipzig Glossing Rules. Additional glosses: PM = patient marker. List of abbreviations: C. = Chinese; E. = English; I. = Italian.

6. References

- Bosc, Franca. 2017. "La Ricetta Nei Blog: Un Coro Di Voci." In *Parole per Mangiare. Discorsi e Culture Del Cibo*, edited by Irina Bajini, Maria Vittoria Calvi, Giuliana Garzone, and Giuseppe Sergio. Milano: LED.
- Brdar-Szabó, Rita, and Mario Brdar. 2009. "Indirect Directives in Recipes: A Cross-Linguistic Perspective." *Lodz Papers in Pragmatics / Special Issue on Speech Actions* 5 (1): 107–31. <https://doi.org/10.2478/v10016-009-0006-x>.
- Byrd, Melanie, and John P. Dunn. 2020. *Cooking through History: A Worldwide Encyclopedia of Food with Menus and Recipes*. Vol. 2. Santa Barbara, CA, USA: ABC-CLIO.
- Cani, Patrizia. 2022. "De-Constructing Recipes: A Constructionist Comparative Analysis of Recipe Texts." MA thesis, Ca' Foscari University of Venice.
- Cotter, Colleen. 1997. "Claiming a Piece of the Pie: How the Language of Recipes Defines Community." In *Recipes for Reading: Community Cookbooks, Stories, Histories*, edited by Anne L. Bower, 51–72. University of Massachusetts Press.
- Diemer, Stefan, and Maximiliane Frobenius. 2013. "When Making Pie, All Ingredients Must Be Chilled. Including You: Lexical, Syntactic and Interactive Features in Online Discourse—A Synchronic Study of Food Blogs." In

- Culinary Linguistics: The Chef's Special*, edited by Cornelia Gerhardt, Maximiliane Frobenius, and Susanne Ley, 53–82. Amsterdam, Philadelphia: John Benjamins.
- Farkas, David K. 1999. "The Logical and Rhetorical Construction of Procedural Discourse. Technical Communication." *Journal of the Society for Technical Communication* 46: 42–54.
- Fontan, Lionel, and Patrick Saint-Dizier. 2008. "Analyzing the Explanation Structure of Procedural Texts: Dealing with Advice and Warnings." In *Semantics in Text Processing. STEP 2008 Conference Proceedings*, 115–127. College Publications.
- Ganier, Franck. 2012a. "Four Characteristics of Procedural Texts." In *Learning to Write Effectively: Current Trends in European Research*, edited by Mark Torrance, Denis Alamargot, Montserrat Castelló, Franck Ganier, Otto Kruse, Anne Mangen, Liliana Tolchinsky, and Luuk van Waes, 307–11. Leiden, The Netherlands: Brill.
- . 2012b. "Some Constraints on the Processing of Procedural Instructions." In *Learning to Write Effectively: Current Trends in European Research*, edited by Mark Torrance, Denis Alamargot, Montserrat Castelló, Franck Ganier, Otto Kruse, Anne Mangen, Liliana Tolchinsky, and Luuk van Waes, 315–19. Leiden, The Netherlands: Brill.
- Garzone, Giuliana. 2017. "Il Genere 'Ricetta' in Lingua Inglese. Dalla Tradizione Ai Food Blog Tra Stabilità Ed Evoluzione." In *Parole per Mangiare. Discorsi e Culture Del Cibo*, edited by Rina Bajini, Maria Vittoria Calvi, Giuliana Garzone, and Giuseppe Sergio, 37–56. Milano: LED.
- Goldberg, Adele E. 1995. *Constructions: A Construction Grammar Approach to Argument Structure*. Cognitive Theory of Language and Culture Series. University of Chicago Press. <https://books.google.it/books?id=HzmGM0qCKtIC>.
- Haiman, John. 1980. "The Iconicity of Grammar: Isomorphism and Motivation." *Language* 56 (3): 515–40.
- Hayakawa, Samuel I. 1972. *Language in Thought and Action*. 3rd ed. New York: Harcourt Brace Jovanovich.
- Hovde, Marjorie. 2022. "Procedural Knowledge and Discourse in Technical Communication: Easy as 1, 2, 3?" In *Assembling Critical Components: A Framework for Sustaining Technical and Professional Communication*, edited by Joanna Schreiber and Lisa Melonçon, 137–69. Colorado: The WAC Clearinghouse and Parlor Press.
- Hsiao, Chi-hua. 2019. "Linguistic Strategies Prompting Interactions in Recipes from Mandarin Chinese Food Blogs." *Text & Talk* 39 (4): 489–510.
- Kaneyasu, Michiko, and Minako Kuhara. 2020. "Regularity and Variation in Japanese Recipes. A Comparative Analysis of Cookbook, Online and User-Generated Sub-Registers." In *Register Studies* 2 (1): 37–71.
- Klenová, Dominika. 2010. "The Language of Cookbooks and Recipes." Masaryk University.
- Lakoff, Robin T. 2006. "Identity à La Carte: You Are What You Eat." In *Discourse and Identity*, edited by Anna De Fina, Deborah Schiffrin, and Michael Bamberg, 142–165. New York: Cambridge University Press.
- Levin, Beth. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. Chicago: University of Chicago Press. <https://books.google.com/books?id=6wIZWOrcBf8C&pgis=1>.
- Levin, Beth, and Malka Rappaport Hovav. 2005. *Argument Realization*. Cambridge University Press. http://books.google.com.au/books/about/Argument_Realization.html?id=msi9a50gHVYC&pgis=1.
- Longacre, Robert E. 1996. *The Grammar of Discourse*. New York and London: Plenum Press.
- Marrone, Gianfranco. 2015. "Buono Da Pensare." In *Cultura e Comunicazione Del Gusto*. Roma: Carocci.
- Norricks, Neal R. 1983. "Recipes as Texts: Technical Language in the Kitchen. In , 73–181. Berl." In *Sprache, Diskurs Und Text*, edited by Rene Jongen, Sabine De Knop, Peter H Nelde, and Marie-Paule Quix, 73–181. Berlin, Boston: De Gruyter.

- Sergio, Giuseppe. 2017. "L'italiano in Cucina." In *Parole per Mangiare. Discorsi e Culture Del Cibo*, edited by Iriwhna Bajini, Maria Vittoria Calvi, Giuliana Garzone, and Giuseppe Sergio. Milano: LED.
- Steels, Luc, ed. 2011. *Design Patterns in Fluid Construction Grammar*. Amsterdam: John Benjamins Publishing Company.
- Trijp, Remi van, Katrien Beuls, and Paul Van Eecke. 2022. "The FCG Editor: An Innovative Environment for Engineering Computational Construction Grammars." *PLoS ONE* 17 (6): e0269708.
- Waxman, Nach. 2004. "Recipes." In *The Oxford Encyclopedia of Food And Drink in America*, edited by Smith Andrew and Bruce Kraig, 347–50. Oxford and New York: Oxford University Press.
- Wharton, Tim. 2010. "Recipes: Beyond the Words." *Gastronomica* 10 (4): 66–73.
<https://doi.org/10.1525/gfc.2010.10.4.67>.This.

Chapter 7

Pragmatics as the Secret Ingredient: a study of reference tracking and evolutive anaphora

Pragmatics as the secret ingredient:

a multilingual study of reference tracking and evolutive anaphors in culinary narratives

Abstract

Anaphora resolution, namely the correct interpretation of anaphoric means, poses numerous challenges for NLP/NLU technologies, particularly when approached from a multilingual perspective. Reference tracking mechanisms exhibit considerable crosslinguistic variation, with some languages heavily relying on inference and common sense. Additional complexities are introduced by phenomena such as zero anaphora, partial coreference, and evolutive anaphors. Evolutive anaphors are expressions that refer to an entity which, in the interim, has undergone changes in its properties or attributes. Crucially, these phenomena are frequent in procedural texts such as recipes, where precise coreference disambiguation is essential for the successful execution of tasks in a technologically driven culinary landscape.

Against this background, the present paper delves into the complexity and variation of reference tracking across languages, shedding light on the challenges faced by NLP and parsing systems. Its aim is twofold: first, to provide an overview of coreference encoding and resolution from a multilingual perspective, with a focus on zero and evolutive anaphors. Second, it seeks to unravel the intricacies of anaphoric encoding in culinary narratives through a study of the type, frequency, and nature of anaphoric devices used in authentic recipes. The paper considers texts in English, Italian, and Chinese, which differ substantially in terms of coreference tracking systems and exhibit varying degrees of reliance on inferential processes for disambiguation.

Keywords: anaphora resolution, null pronouns, evolutive anaphors, multi-language perspective, inference

1. Introduction

In the realm of natural language processing (NLP), recipes present a fascinating and challenging domain. Activities in the kitchen environment are particularly complex, usually involving many objects and interactions between them at different stages. Ingredients undergo rapid and substantial changes of state, getting moved and transformed throughout the process. Each step frequently leads to the creation of new items (resultant objects, see Morbiato and Cani, this volume), which are then subject to further movement and transformation, evolving over time as each subsequent step is carried out. Achieving a precise comprehension of recipe texts becomes pivotal in a technologically driven culinary landscape, especially with respect to when and how to use specific items during the various steps of preparation. Crucially, this comprehension comes with its set of challenges. Recipe texts abound with references to contextual items, including ingredients and kitchen tools, presupposing solid awareness and mastery of world knowledge and common sense. Moreover, they frequently incorporate multiple modalities, encompassing not only the text (the recipe itself) but also images (depicting ingredients and the cooking process) and, at times, audio, or video elements. Third, the linguistic means used in recipes to refer to items, ingredients, and tools are sometimes far from transparent and often include weak anaphoric means, such as pronouns or even zero anaphors (henceforth signalled as $\emptyset$). Ingredients and 'resultant objects' (i.e., items that are created as a

result of a specific step, as e.g., a dough, or a puree, cf. Morbiato and Cani, this volume) are frequently left unmentioned, as exemplified in phrases like 'Bake $\emptyset_x$ until $\emptyset_x$ crispy and $\emptyset_x$ golden' or 'Pour $\emptyset_y$ until $\emptyset_y$ saturated', resulting in potential ambiguity. This phenomenon is not marginal; in fact, zero anaphors (also referred to as zero pronouns, null elements, or implicit entities) are frequent in procedural texts such as recipes, particularly when compared to other genres. Further challenges include partial coreference phenomena (also known as bridging) as well as so-called evolutive anaphors, namely linguistic expressions that refer to an entity while also indicating a change in its properties or attributes over time. Evolutive anaphors are common in culinary narratives, where they typically refer to a previously mentioned ingredient or component of the recipe which, in the meantime, has undergone some transformation. For example, in (1) the pronoun *it* is an evolutive anaphor, involving a change in properties of the pasta, from being hard to being soft after cooking. Reference tracking might be even more challenging when a zero is used instead of an overt pronoun to express evolutive anaphors, as in (2). Crucially, inference plays a significant role in coreference disambiguation, in addition to, or in place of, lexical and grammatical encoding. In fact, it is world knowledge and common sense that enable the reader to understand that the anaphors in (1) and (2) refer to *pasta*, and not to *pot* or *water*.

- (1) Boil a pot of water and add the **pasta**. Once **it**'s cooked, drain **it** and then add **it** directly to the tomato sauce.
- (2) Boil a pot of water and add the **pasta**. Once $\emptyset$ cooked, drain $\emptyset$, and then add $\emptyset$ directly to the tomato sauce.

The study of anaphora and in-text referencing spans linguistic fields like semantics, syntax, and discourse, and intersects with disciplines like semiotics, psycho- and neurolinguistics, and cognition. Undoubtedly, it is also crucial for both theories and applications of NLP due to the challenges inherent in anaphora identification and resolution. These challenges become particularly pronounced in multilanguage approaches, as reference tracking mechanisms exhibit considerable crosslinguistic variation. Typologically distant languages employ different devices, which may be predominantly lexical, morphosyntactic, but also inferential in nature. Crucially, procedural texts, such as recipes, contain rich anaphoric phenomena, yet they have been relatively overlooked in the NLP field, with existing research primarily focusing on anaphora resolution in declarative texts like news articles (Yamakata, Mori, and Carroll 2020; Fang, Baldwin, and Verspoor 2022).

Against this background, the primary objective of this paper is to unravel the complexities of non-transparent anaphoric references in recipes from a multi-language perspective. More specifically, it provides an in-depth exploration of coreference encoding and resolution, with a particular emphasis on zero pronouns, partial coreference phenomena, and evolutive anaphors. Additionally, it presents an extensive analysis of the type, frequency, and nature of anaphoric devices used in authentic recipes drawn from international blogs. The paper considers texts in English, Italian, and Chinese, which significantly vary in their coreference tracking mechanisms and demonstrate different levels of dependence on zero anaphors and inferential processes.

The rest of the paper is structured as follows. Section 2 provides a cross-linguistic overview of reference encoding and tracking mechanisms, with a focus on zero pronouns, partial coreference, and evolutive anaphors, with some remarks on the challenges involved for NLP. Section 3 analyses the correlation between the structure of recipes, information flow, and grammatical encoding of anaphoric means. Section 4 presents the study and its results. Lastly, section 5 draws some conclusions.

2. Anaphora, reference encoding, and tracking

When talking about sequences of situations in which the same participants are involved, it is necessary to refer to them in each clause in such a way that they can be identified as being the same as or different from the participants referred to in previous clauses (Foley and Van Valin 1984, 1). This helps granting textual cohesion, which concerns the way various elements within a piece of writing or speech are linked together to create a unified and understandable whole. More precisely, reference relations may be either with entities existing in the context to which the utterance refers (e.g., the kitchen environment) or with elements present in the preceding text (e.g., the recipe). This distinction is articulated by Halliday and Hasan (1976) in the concepts of 'exophora' and 'endophora'. Exophora occurs primarily in texts where non-linguistic factors, such as the situational context, play a dominant role in semantic terms, with language serving as a secondary component. According to the authors, cooking recipes can be considered examples of such texts, as they exhibit rich contextual references (1976, 24). Endophora, conversely, refers to in-text reference relations, like anaphora (including cataphora and zero anaphora).

The term 'anaphora' is derived from the Greek word ἀναφορά, a compound word consisting of ἀνα 'back, upstream, back in an upward direction', and φερα 'the act of carrying', denoting the act of carrying back upstream. It refers to a relation between two elements, wherein the interpretation of one (generally called 'anaphor') is determined by the interpretation of the other (generally called 'antecedent'). In (3), the antecedent *onions* is then being referred to through weaker anaphoric means, i.e., the pronouns *them* and *they*.

(3) Chop the **onions** finely. Then, sauté **them** in a pan until **they** turn golden brown.

Crucially, each language differs in the range of possibilities for creating anaphoric relations between elements in texts, which may be expressed through both morphosyntax and lexicon (Halliday and Hasan 1976, 34). Moreover, languages exhibit considerable variation in the extent to which grammar and inference play a role in coreference disambiguation. Awareness of such differences is paramount in NLP approaches that adopt a multilanguage perspective. In some languages, disambiguation is facilitated through lexical information or syntactic structure, while in others, it heavily relies on inferential processes involving context, common sense, and real-world conceptual knowledge. Cross-linguistic variation observed in this domain has led scholars like Foley and Van Valin (1984) to outline a typology of reference tracking systems, which will be introduced in Section 2.1. Furthermore, the choice among different anaphoric encodings (noun repetition, substitution with hypernyms or synonyms, pronouns, or zeroes) is not random: the use of a pronoun over noun repetition may provide opposite instructions towards antecedent retrieval. This aspect will be illustrated in Section 2.2. Sections 2.3 and 2.4, finally, present a focus on anaphors that pose a particularly significant challenge for NLP—namely, zero, partial coreference, and evolutive anaphors.

2.1. Reference tracking from a multi-language perspective

In linguistics, reference tracking systems refer to the mechanisms employed in different languages to keep track of the various entities mentioned in an ongoing discourse. Typological approaches to language description typically identify four major types of reference-tracking systems, which a given language may use one or more of: (i) gender systems, (ii) switch-reference systems, (iii) switch-function systems, and (iv) inference systems (Foley and Van Valin 1984; Comrie 1989; Huang 2000). The first type is commonly considered lexical, the second and third are grammatical, and the fourth is pragmatic. Italian, along with many other genetically different languages, and to a lesser extent English, are examples of the gender/class system: nominal phrases are gender/person/number marked, hence further mentions may be tracked through gender marking on pronouns or verbal affixes or by employing a combination of both strategies. While this phenomenon is observable to a limited extent in English (4), Italian's gender and number

It has often been observed that variations in the encoding of anaphoric relationships pose challenges for NLP applications, such as Machine Translation: for example, when translating from and into languages which mark the gender of pronouns, resolving anaphoric relations becomes crucial (Mitkov 1999; 2022). According to Mitkov, additional complexity arises from gender or number discrepancies in words denoting the same concept, differences in the selection of anaphors in the target language, and ellipsis of pronouns in both the source and target languages.

2.2. Choice of anaphoric means

It is widely accepted that the choice among different anaphoric devices is far from random and is in fact vital for allowing correct reference tracking (Kintsch and van Dijk 1978; van Dijk and Kintsch 1983; Levinson 2000). Choosing a full noun phrase over a pronoun, for example, cues the reader to search for the reference beyond the nearest context:

- (8) a. **Chopin**_i finished playing, hence **he**_i exited the concert hall.
 b. **Chopin**_i finished playing, hence **the man**_k exited the concert hall.

In (8) (a), the pronoun *he* clearly refers back to Chopin. Conversely, the stronger anaphoric reference in (b), i.e., *the man* (a hypernym), may feel marked (in that unnecessary), and lead the reader to interpret it as some other person mentioned earlier in the text (cf. Levinson (1987; 2000) and subsequent literature).

The choice of anaphoric means has been shown to adhere to specific discourse pragmatic principles, rooted in the concept of information flow (see Morbiato (2022) on this notion), which captures how noun referents are introduced in discourse based on their information status (Chafe 1976; 1994; Du Bois 1980; Givón 1983). According to Chafe (1994), a referent's information status can fall into three categories: new, accessible, or given. When a referent is new, it resides in the periphery of the hearer's consciousness, making it less accessible. Consequently, the speaker must introduce it explicitly, e.g., by using a full noun phrase, such as 'John Smith'. When a referent is accessible, is more active in the listener's consciousness, possibly because it has been already mentioned, though not in the immediate discourse. In such cases, a full noun phrase is still required, though an abbreviated version like 'John' may suffice. In contrast, a given referent is already active, often due to recent mentions. Consequently, it is highly accessible, and a pronoun like 'he', or even a zero anaphor, can be employed instead of a full noun phrase. Similarly, Givón argues that the choices, made by speakers, of using one referential device over the others depend on the degree of referential or topic continuity, i.e., the possibility to identify the referent, in that it has been recently mentioned or is contextually present or accessible. According to this principle, the more continuous and predictable a discourse topic is, the more accessible it becomes to the reader or hearer, thus requiring less overt expressions. Givón defines a hierarchy of cohesive means ranked in terms of their degree of referential continuity.

- (9) Referent-coding devices and referential continuity (Givón 1983)
- > *lowest referential continuity*
 - a. indefinite NPs
 - b. definite NPs
 - c. stressed independent pronouns
 - d. unstressed anaphoric pronouns
 - e. zero anaphora
 - > *highest referential continuity*

The physical distance from the last mention of a referent plays a role in determining the appropriate anaphoric form. When the distance is short, a reduced anaphoric form is more likely to be used, in that the antecedent remains readily accessible to the hearer, as in (8) (a). Punctuation may also play a role. In (10) below, the unstressed anaphoric pronoun in sentence (a) and zero anaphor in sentence (b) both convey the idea of maximal referential continuity, thus suggesting the hearer to trace the referent back to the subject of the first clause, 'John'. However, sentence (b) represents an unsuitable continuation because, in English, zero anaphora can only be employed within chain-medial junctures, and not across chain boundaries (Givón 1983, 9).

- (10) John went to the mirror, Ø examined his hair, Ø sighed and Ø turned.
 a. Then he walked out.
 b. *Then Ø walked out.

Romance languages like Spanish and Italian were also observed to follow the hierarchy in (9); however, as morphologically rich, gender-system languages (cf. Section 2.1), they feature systematic pronoun subject dropping, obligatory pronominal agreement, and unstressed clitic object pronouns (11).

- (11) Non Ø ho invit-ato Gianni perché Ø non lo cono-sco. (W. Li 2005)
 NEG have.PRS.1SG invite-PTCP because NEG 3SG.M.ACC know-PRS.1SG
 'I didn't invite Gianni because I don't know him.'

Inference-system, East-Asian languages like Chinese, on the other hand, are well-known for making a very large use of zero anaphora, which is also employed in cases of significant distance between the anaphor and its antecedent (Givón 1983, 11–13; L. Tao and Healy 2005; L. Tao 2001). Zeroes can appear in different syntactic slots (not only subject, like in Italian) and can cross both sentence and paragraph boundaries (12). What in English is mostly done by pronouns, in Chinese is done by zero anaphors (Morbiato 2015). Using an overt pronoun instead of a zero anaphor may instruct the reader to locate the antecedent in the distant text. Furthermore, in natural conversations antecedents remain accessible even if separated by other intervening referents (L. Tao 1996):

- (12) (a) 片子取来了。 (b) 孙主任看了, (c) 在场的医生都轮流看着。 (W. Li 2005)
 (a) piànzi_i qǔ-láile. (b) Sūn zhǔrèn kàn-le Ø_i
 X-ray bring-over-PFV/COS Sun dr. look-PFV
 (c) zàichǎng de yīsheng dōu lúnlú kàn-zhe Ø_i
 present DET doctor all turn look-DUR
 'The X-ray was obtained. Dr. Sun examined (it). All the other doctors took turns to also examine (it).'

Both information flow and topic continuity theories offer a comprehensive framework for understanding reference management and have garnered support from cross-linguistic studies using natural discourse data. Psycholinguistic research has also confirmed this, relating the use of referential devices to the cognitive aspects of memory and attention. Using a repeated name instead of a pronoun in consecutive sentences disrupts discourse coherence and prolongs processing time, a phenomenon known as the "repeated name penalty" (Gordon, Grosz, and Gilliom 1993). However, other research suggests that proximity and interference are not the sole factors influencing the choice of pronouns: in English discourse, too, a pronoun may be used even in the absence of an immediate antecedent (Greene et al. 1994). In such cases, a shared common ground between the speaker and hearer can activate the reference, rendering the pronoun accessible.

2.3.Zero anaphora: challenges for parsers and NLP

Zero anaphora involves an empty grammatical slot representing a nominal referent in a sentence, devoid of lexical or grammatical markings. Despite its seemingly non-transparent nature, zero anaphora is considered “one of the most natural, universal, ancient and functionally-coherent grammatical devices in the tool-kit of natural language” (Givón 1983). The rationale behind zero anaphora lies in communicative principles tied to information predictability and salience. Predictable and non-salient information need not be explicitly mentioned. While zero anaphora is prevalent in all mature grammars, inference-system languages like Chinese, Korean, and Japanese exhibit even higher instances. Failing to detect and interpret zeroes in these languages can significantly impede comprehension.

The rules governing the licensing of zero anaphora show considerable variations among languages. In English, zero anaphora is permitted in subject positions when the subject corefers with that of the previous clauses, while objects must be overt (13) (E). In contrast, Italian requires the omission of all pronominal subjects without the need for prior mention; objects are obligatorily overt, either as stressed or as clitic pronouns, as *la* in (13) (I). In Chinese, any verbal argument can be represented as a zero as long as it encodes given information (13) (C). Zero anaphors play a crucial role in maintaining discourse cohesion and contribute to the formation of 'topic chains' (W. Li 2005).

(13) E. He dressed up, $\emptyset_j$ went down the stairs, $\emptyset_j$ walked out **the front door** and $\emptyset_j$ closed **it** behind him.

I. $\emptyset_j$ Si vestì, $\emptyset_j$ scese le scale, $\emptyset_j$ uscì dalla **porta d'ingresso** e **la** chiuse alle sue spalle.

C. 他穿好衣服，走下楼梯，走出前门后就顺手关上了。

tā_j chuān-hǎo yīfú, $\emptyset_j$ zǒu-xià lóutī_k,
3SG dress-good cloth walk-down stairs

$\emptyset_j$ zǒu-chū qián-mén hòu $\emptyset_j$ jiù shùnshǒu guān-shàng-le $\emptyset_k$
walk-exit front-door after then smoothly close-up-PFV/COS

As common as it may be, zero anaphora (ZA) remains a significant challenge for NLP. A first major issue is ZA detection. In the context of parsers, a probabilistic approach is generally adopted, with many treebanks employing empty nodes in parse trees to represent omitted elements (Chung and Gildea 2010; Bies et al. 1995; Xue and Xia 2000). Nevertheless, as parsers commonly limit the probabilistic engine from postulating the existence of zero anaphora (ZA) everywhere, the machine's representation of sentential meaning frequently becomes inconsistent (Delmonte 2009; 2014). Specific challenges emerge in inference-system East-Asian languages, where zero anaphora (ZA) is prevalent. In the Chinese Treebank, zeroes appear approximately once in every four sentences on average. This results in varying degrees of accuracy levels in syntactic parsers, especially in subject- or topic-drop languages such as Italian and Chinese. In more recent, deep learning based approaches, zero anaphora detection is mainly accomplished through two methods: syntactic analysis of sentences or binary classification of predicates, with varying degrees of accuracy (Kim, Ra, and Lim 2021, 300; Iida and Poesio 2011). The second challenge is Zero Anaphora Resolution (ZAR). Early studies used rule-based frameworks, but with the increasing availability of annotated corpora, statistical and probabilistic models gained prominence. Later, various machine-learning models were developed, while recent trends lean towards deep learning approaches that aim to automatically learn feature representations (Lata, Singh, and Dutta 2021), where words can be represented in a vector space (Word Embedding) that provides both semantic and syntactic information. Chen and Ng (2016) are among the first to develop a neural network-based approach incorporating both word embedding and hand-crafted features. Following studies, mainly focusing on East Asian languages, have made significant improvements, aiming to account for the long-term influence of antecedents (Yin et al. 2018; Iwata, Watanabe, and Nagata 2021; Kim, Ra, and Lim 2021; L. Song et al. 2020). However, "the performances of best systems up to now are not high even after a long history of research" (Kim, Ra, and Lim 2021, 300). While anaphora resolution is crucial for NLP applications, if it cannot be achieved with sufficient precision, it could pose far more problems than it could solve (Lata, Singh, and Dutta 2021,

54:2922). Mitkov (2022, 276) also concludes that there are outstanding issues, including the performance limits of anaphora resolution algorithms and of knowledge-poor methods. Pre-processing accuracy remains suboptimal, affecting system performance. Most anaphora resolution systems require manual pre-processing or pre-analysed corpora, and only a few operate fully automatically. Finally, Mitkov underscores the need for further research in multilingual contexts.

2.4. Partial coreference, bridging, and evolutionary anaphors

A further challenge in machine representation involves partial coreference, commonly known as 'bridging.' This anaphoric phenomenon occurs when an anaphor is interpreted based on a non-identical associated antecedent, relying on background assumptions or inferences (Clark 1977; Clark and Haviland 1975; Huang 2000; Asher and Lascarides 1998). According to Clark,

Bridging [...] is an obligatory part of the process of comprehension. The listener [...] needs] to identify the intended referents (in memory) for all referring expressions. [...] In most instances, the success of this strategy requires the listener to bridge, to construct certain implicatures, and so he takes these implicatures too as a necessary part of comprehension. In short, he considers implicatures to be intrinsic to the intended message, since without them the utterance could not refer. (Clark 1977, 413)

The bridging relationship may have different semantic facets, as illustrated below with examples in the cooking domain:

- (14) Lisa opened **the picnic basket**. **The sandwiches** were perfectly crafted. (Setting-element)
- (15) Cut **the lemon** and take **the seeds** out. (Whole-part/necessary parts)
- (16) Slice **the bread** into **pieces**. (Partitive)
- (17) Add all **dry ingredients**, except **yeast**. (Set-membership)
- (18) Add all **fruits**, except **strawberries**. (Hypernym-hyponym)
- (19) Bake for 15 minutes, then turn **the oven** off and let it cook. (Necessary role/tool)

Bridging cross-reference anaphora is most commonly encoded with definite NPs (Huang 2000, 3:7). However, in recipes other instances are possible, such as with numerals, quantifiers, and measure words, and indefinite adjectives/pronouns (highlighted in bold):

- (20) Remove sinew and membrane from the chicken breast, chop, or mince very finely to almost a paste. Mix with $\frac{1}{2}$ **a tablespoon of the corn flour**, milk, add the egg white and blend well. [...] With one chopstick pick up **a little of the chicken mixture** and drop that in the stock, go on until **all the chicken** is used up and you have **lots of chicken bits/balls** pea size. (Fang et al. 2022)

A further complexity is constituted by evolutive anaphors, which referring to entities undergoing transformation, evolution, or gradual changes over time. These anaphors are frequently encountered in scientific, technical, and procedural texts, including recipes, but can also be found in everyday language. Asher (2000) is credited as one of the first linguists to introduce the concept of evolutive anaphora, even though its existence had previously been observed by Brown and Yule (1983) in recipe texts.

- (21) Wash and core six cooking apples. Put **them** onto a fireproof dish. (Brown and Yule 1983, 201)

As Brown and Yule observe, the anaphora *them* does not properly refer to *six cooking apples*, as they have undergone a change of state, i.e., they are now washed and cored. Sometimes, the antecedent is not overtly expressed, as in (22). Clearly, such anaphors are even more challenging for parsers to handle, especially when the anaphoric mean is a zero (23):

(22) Add tomatoes and simmer for 30 minutes [...] Combine pasta and **sauce**_i.

(23) Add tomatoes and simmer for 30 minutes [...] Combine **Ø**_i with pasta.

In 1983, Brown and Yule already emphasized the delicate nature of reference in this type of anaphora and proposed the need for a model of interpretation that allows textual referents to accumulate features or undergo changes as the discourse progresses:

The processor establishes a referent in his mental representation of the discourse and relates subsequent references to that referent back to his mental representation, rather than to the original verbal expression in the text. (Brown and Yule 1983, 200–201)

Korzen, too, devoted particular attention to evolutive anaphora, which he defines as “the context in which a textual referent is referred to through an anaphoric expression after undergoing radical transformations in its ontological state” (2009, 323, our translation). He specifically looked at such anaphors in Italian recipes:

(24) Lessate **le patate**_i, sbucciate-**le** e passate-**le** calde dallo straccio sopra a un velo di farina. Fate una buca sul monte **delle patate**_k, salate**le**_k, date **loro**_k l'odore della noce moscata [...].

‘Boil the potatoes, peel them, and pass them hot from the tin over a layer of flour. Make a hole in the mound of potatoes, salt them, give them the smell of nutmeg [...].’

In (24), the potatoes designated by the antecedent *le patate*_i then become mashed. Despite undergoing a significant transformation, the referent is then recalled similarly with the PP *delle patate*_k ‘of the potatoes’. Korzen argues that this continuity arises from the reader recognizing a certain homogeneity in the natural entity that has undergone a change. Although the entity experiences a modification in its ontological state, its main properties, such as color, flavor, and smell, which directly influence the recipe, remain unchanged, as illustrated in Figure 1:

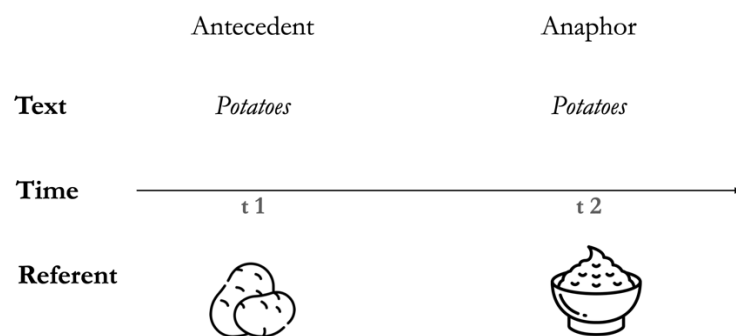


Figure 1: Evolutive anaphor (elaborated from Korzen 2009)

Similarly, in the English recipe below, the referent of the noun *potatoes* undergoes significant changes in the various steps, although being referred to with the same noun throughout the text:

(25) Bring a large pot of salted water to a boil. Add **potatoes** and garlic, lower heat to medium, and simmer until **potatoes** are tender, 15 to 20 minutes. When the **potatoes** are almost finished, heat milk and butter in a small saucepan over low heat until butter is melted. Drain **potatoes** and return to the pot. Slowly add warm milk mixture, blending it in with a potato masher or electric mixer until **potatoes** are smooth and creamy.

However, acceptability may vary based on the type of transformation and within certain limits, known as 'ontological thresholds' (Korzen 2009). A critical threshold involves the change of state from solid to liquid, which is so significant that a new form (the juice) is necessary (26) (Figure 2):

- (26) a. *Juice **the apples**, then put **them** onto a pan.
b. Juice **the apples**, then put **the juice** onto a pan.

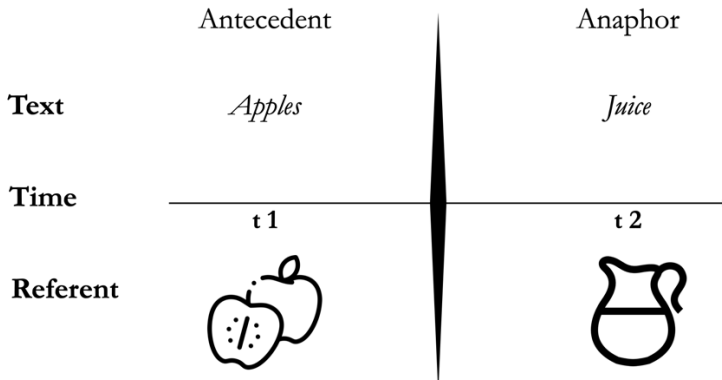


Figure 2: *Evolutionary anaphor - threshold*

Evolutionary anaphors constitute a significant challenge in NLP, especially when encoded through zero pronouns, as in (2) and (23), and when multimodal anaphora resolution is involved. For example, Oguz et al. (2022) look at recipes with videos or images incorporated: in Figure 3, *the cubes* is used to describe the bread pieces in the text because the instruction *chop the bread* shaped them into cubes in the video:



Figure 3: *Examples from the YouCookII dataset (from Oguz et al. 2022)*

Recent progress has been made in addressing processes as well as bridging and evolutionary anaphora in procedural text understanding. One prominent research direction involves representing procedures as workflow graphs structured around actions, with nodes representing named entities and edges representing relational structures, effectively capturing ingredient interactions. Yet, the approach falls short in addressing generic anaphora phenomena. Bosselut et al. (2018) predict action and entity attribute dynamics in recipes using a recurrent memory network, yet they do not directly tackle zero-anaphora. The RISEC corpus (Jiang et al. 2020) focuses on identifying zero-anaphora verbs in English recipes but relies on manual annotation and does not fully address general anaphoric phenomena. Some studies consider evolutionary anaphors as instances of bridging: Fang, Baldwin, and Verspoor (2022) propose an annotation framework for bridging anaphors applied to the RecipeRef corpus. However, as noted by Oguz et al. (2022),

"temporally evolving entities present rich and, to date, understudied challenges for anaphora resolution," which have not yet been thoroughly addressed.

3. Information flow and anaphoric choice in recipe texts

As observed, anaphoric coreference in recipes is not transparent, heavily relying on zero anaphors rather than explicit nouns. The agent, i.e., who performs for all actions and processes, is typically left unspecified. Tracking coreference for patients, the entities undergoing these actions and processes, is also complex. In recipes, both bridging and evolutive anaphors are common as ingredients (and specific parts or quantities of them) undergo swift and substantial changes. This leads to the creation of new entities, i.e., resultant objects, which, in turn, undergo further movements and transformations. Effectively addressing the intricacies of reference tracking in recipes requires breaking down complexity into more manageable components. To start, let us examine how information is organized within recipes. The structure of recipes significantly influences the flow of information in the text, thereby impacting the choice of anaphoric means.

A typical recipe consists of two essential components: a comprehensive list of the required ingredients and a detailed set of instructions outlining the steps for food preparation (Norrick 1983). Since the ingredients list comes first, in subsequent mentions, these items become activated and highly accessible in the reader's mind (Chafe 1976). In contrast, the instructions primarily focus on the actions needed to transform the ingredients into a finished product and how to use the necessary kitchen tools for the task. In this component, ingredients are still perceived as accessible/given, whereas the new, most salient information revolves around the action and its result, including possible result expressions (cf. Morbiato and Cani, this volume). Within the instructional compound, it is important to note that all mentioned items are anaphoric. These may bear either exophoric reference, namely pointing to objects in the situational context, (e.g., tools like the fridge or blender), or endophoric reference, which encompass above-listed ingredients. At the beginning of the instructional compound, patients are generally chosen within the list of ingredients. As a consequence of the actions expressed by verbs, resultant objects are created. As the recipe progresses, the resultant object of a specific step is the patient of the action in the following step, as Figure 4 shows:

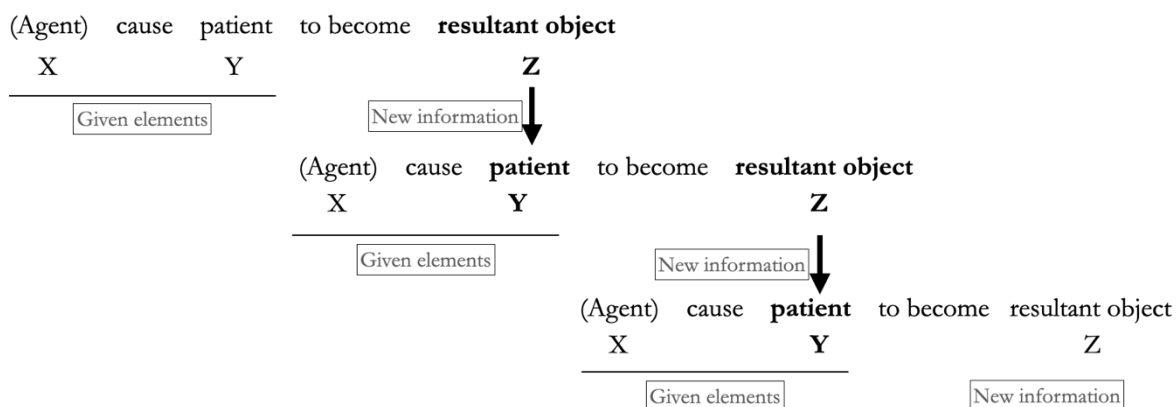


Figure 4: Argument structure and information flow in recipe texts

However, this is not always the case: processed entities in recipes can be mentioned far later in the text (especially in modular recipes, as e.g., in recipes where a cake, cake filling, and cake icing are separately prepared, and only combined in a final step).

The activation status of the various antecedents/anaphors is reflected in the grammar and in the choice of anaphoric means. The non expression of agents is reflected in the pervasive use of the imperative mode (Norrick 1983, 176). English recipes employ agency-neutral and time-neutral imperative sentences that lack an overt subject, as the verb is clearly directed at the reader (Wharton 2010, 68). Similarly, objects, which typically encode patient on a syntactic level, are also frequently realized through zeroes. This practice is observed even in languages like English and Italian, where the omission of objects is generally not observed or allowed. Bender (1999) and Massam & Roberge (1989) investigate the occurrence of null objects in English recipes, linking the phenomenon to this specific genre. Massam (1987) investigates similarities between construction where the patient is encoded through zeroes, i.e., middle (27) (a), tough (27) (b), and null-object constructions (27) (c):

- (27) a. This bread cuts Ø easily.
 b. This bread is easy to cut Ø.
 c. Take bread. Cut Ø carefully (and place Ø in pie plate) (Massam 1987:1)

Italian, similarly, also makes extensive use of direct object deletion in recipe texts and this feature appears to be strictly related to the textual genre of recipes. The phenomenon of zero anaphora is even more accentuated in languages like Chinese. Topic chains, as well as subjectless and semantically passive sentences, serve a functional role in recipes by presenting the sequential procedure more effectively and concisely. This approach allows the reader to concentrate on the patient and its transformations. In both sentence types, the emphasis is predominantly on the action and its outcome (H. Tao 2007).

4. The study: research questions and methodology

Building upon the background laid out so far, this study endeavours to provide a detailed analysis and comparison of anaphoric phenomena in original recipes across English, Italian, and Chinese. The analysis seeks to identify the types and frequencies of anaphoric devices employed in the texts, paying particular attention to the selection between different forms (e.g., full noun phrases over pronouns or zero forms). Additionally, the study examines the nature of anaphoric relations, distinguishing between exact or partial coreference, evolutive anaphors, and other relevant categories. The second research objective aims to elucidate the disambiguation process and assess the extent to which inference is necessary for successful anaphora resolution.

As in Morbiato and Cani (this volume), the dataset comprises recipes from food blogs (somewhatsimple.com, pandipane.blogspot.com, and meishichina.com). As typical blog entries, the texts are also correlated with pictures depicting each step, additional paragraphs that present the dish, a detailed list of ingredients (sometimes with possible variations) followed by a blog-like discussion about the procedure and pictures showing the cooking steps and, in some cases, a section called 'small tricks' and comments from the readers (cf. Morbiato and Cani, this volume for a discussion of blog-like recipe texts). Hence, multiple modalities are involved: understanding of procedures is aided by both pictures and non-procedural textual parts, while cohesion is achieved not only through endophoric devices but also through contextual reference and meta-textual means. In the texts, each referent is marked with a subscript letter next to it when it is first mentioned. The same letter is then used to denote other referential devices or zero anaphora referring to it. Referents and referential devices are highlighted in bold.

4.1. Types of anaphoric devices

Table 1 shows the anaphoric devices employed in each language. As all recipes involve the use of subjectless imperative sentences, the table does not include zeroes in the subject position. Full NP repetition is frequent in all three languages and so is zero anaphora, especially in Chinese, where pronoun use is absent. Prepositional phrases are used almost solely to introduce exophoric reference, specifically tools that a kitchen is supposed to be equipped with (e.g., *in a large bowl*, *into a blender*, *in forno* 'in the oven'). In Chinese, exophoric means may consist of nouns plus localizers, as *chúshíjī-nèi* 'mixer-in' in (C.10), or also bare NPs, as *zhùzhījī* 'juicer' in (C9) (b) below. In any case, exophora is always expressed with overt means, be them NPs or PPs. Endophora, and specifically reference to ingredients and resultant objects, is encoded both through full repetition and weaker anaphors, although with significant differences among languages as illustrated in Table 1. Overall, Chinese makes the most extensive use of zero anaphora which, however, is very frequent in English, as well. Italian, on the other hand, maxes both use of zeroes and of clitics.

Table 1: Anaphoric devices distribution

Referential devices	English	Italian	Chinese
Full NPs	33	25	22
PPs/nouns + localizers	11	15	7
Stressed Pronouns	2		
Unstressed/clitic pronouns		8	
Zero anaphors	16	8	28
Word count	329	321	411 (characters)

English employs both full NPs and zero anaphora both for patient arguments and for resultant objects, as the two zeroes in (E.74) show, while pronouns are scarce. Interestingly, repetition of full NPs is used regardless of referents' activation stages, almost on par with zero anaphora. This is evident in (E.76), where the noun *dough* occurs twice despite it being highly activated and encoding exact coreference. Zero elements are typically recoverable from the text anaphorically, although some use of cataphora is also present. For instance, in (E.80), the zero ($\emptyset_s$) refers to 'rolls' mentioned later on in (E.82).

(28) (E.74) Knead $\emptyset_o$ 5-10 minutes, until $\emptyset_o$ smooth.

(E.75) Place $\emptyset_o$ in a greased bowl and let $\emptyset_o$ rise one , or until $\emptyset_o$ doubled.

(E.76) When the **dough** has risen, gently punch the **dough** down and divide $\emptyset_o$ into two equal sized **balls**_p.

(E.77) Roll each **ball** out into a **circle**_q that is approximately 18 inches in diameter and spread $\emptyset_q$ with two tablespoons of the remaining butter.

(E.78) Using a pizza cutter, cut the **dough** into twelve equal sized **wedges**_r.

(E.79) Roll up $\emptyset_r$, starting from the wide edge, into a **crescent shape**.

(E.80) Place $\emptyset_s$ on a greased or parchment lined baking sheet, and repeat with remaining **dough**.

(E.81) Roll the second ball of **dough** out, and repeat the process.

(E.82) You should be able to fit all **24 rolls**_s on one large baking sheet.

In Italian, coreference is primarily encoded through full NPs, either repeated, as e.g., the word *impasto* 'dough' in (I.16), (I.17), and (I.19), or through synonyms/hypernyms: *polpa di zucca* 'pumpkin pulp' in (I.14) points to the same referent as *purea* 'pumpkin puree', which was used at the beginning of the procedure (*Prepar-ate la purea di zucca* 'Prepare the pumpkin puree'). While direct object deletion is extremely rare in other genres, it is found here in (I.17), (I.18) and (I.19). Given referents in object position are also encoded through weak, clitic pronouns, e.g., *mettete-la* 'put-it' in (I.17) (b) and *sgonfiando-lo* 'deflating-it' in (I.19) (b). Resultant objects are expressed through expressions like *un composto cremoso*, lit. 'a creamy mixture' (I.14) (c) or *impasto* 'dough' (I.16.) (b), but also with zeroes.

(29) (I.14)(a) Mett-ete in una ciotola **il quantitativo di polpa di zucca_i** (120gr),
put-IMP.2PL in a bowl **the quantity of pulp of pumpkin_i** 120 gr

(b) **la Pasta Madre (a pezzetti), lo zucchero e parte d-ell' acqua_m** (es. 100 gr),
the sourdough in pieces the sugar and part of-the water_m ex.

(c) ora mescol-ate bene **il tutto_{i+m}** ottenendo **un composto cremoso_n**.
now mix-IMP.2PL well **the everything_{i+m}** obtaining **a mixture creamy_n**

'Place 120 grams of pumpkin puree, the sourdough cut into pieces, sugar, and a portion of the water (e.g., 100 grams) in the bowl. Mix everything thoroughly to achieve a creamy texture.'

(I.15)(a) Aggiung-ete la farina e gradualmente inseri-te **la restante acqua_n**.
add-IMP.2PL the flour and gradually insert-IMP.2PL **the remaining water**

(b) può darsi che non serva **tutt-a_n**.
might be that NEG need **all-F.SG**

'Add the flour and gradually incorporate the remaining water; you might not need all of it.'

(c) ... (d) ...

(I.16)(a) Quando **Ø_o** sarà incordato aggiung-ete a filo l' olio di soia,
when **Ø_o** be blended add-IMP.2PL at thread the oil of soy

(b) **impast-ate fino.a.che l' impasto_p** non sarà omogeneo, liscio ed elastico.
knead-IMP.2PL until **the dough_p** NEG be.FUT homogeneous smooth and elastic

'When the dough is blended, slowly add some soy oil. Knead until the dough becomes homogeneous, smooth, and elastic.'

(I.17) (a) Pirl-ate un po' l' **impasto_p** a formare **un-a bell-a palla lisci-a_q**
spin-IMP.2PL a bit the **dough_p** to form **a-F nice-F ball.F smooth-F_q**

(b) ole-ate **Ø_p** leggermente e mett-ete-**la_p** in una ciotola ...
oil-IMP.2PL **Ø_p** slightly and put-IMP.2PL-F.3SG_p in a bowl ...

'Give the dough a gentle spin to form a nice, smooth ball. Lightly coat with oil, place in a bowl and cover.'

(I.18)(a) Mett-ete **Ø_q** ora in frigo per una decina di ore
put-IMP.2PL **Ø_q** now in fridge for a ten of hours

'Place it in the fridge for approximately ten hours.'

(I.19)(a) Tolt-**Ø_i** da-l frigo fa-te tornare **Ø_i** a temp. ambiente per un' ora,
removed-M.SG_i from-the fridge make-IMP.2PL return **Ø_i** at temp. room for one hour

(b) stend-ete con il mattarello **l' impasto_m** sgonfiando-**lo_m** bene ma delicatamente,
spread-IMP.2PL with the rolling pin **the dough_m** deflating-3M.SG_m well but gently

(c) tirate **l' impasto_n** a-ll' altezza di circa 5 mm

stretch-IMP.2PL **the dough_n** at-the height of about 5 mm
 ‘Take the dough out of the fridge and allow it to come to room temperature. Roll out the dough with a rolling pin, gently deflating it, and stretch it until it reaches around 5 mm in height.’

As expected in a gender-system language like Italian, anaphoric resolution is mainly achieved through clitic pronouns and gender/number agreement. For example, in sentence (I.19) (a), the reference of the masculine past participle verb *tolt-o* ‘removed’ (masculine) is *impasto* (masculine), which is mentioned cataphorically in the clause that follows, and not the closest NP *palla* (I.17), which is feminine and linked to the clitic *-la* in *mett-ete-la* ‘put it’ (I.17) (b). However, inference plays a crucial role as well: the NP *tutta* ‘entire amount of’ in sentence (I.15) refers to the word *acqua* ‘water’, and not to the word *farina* ‘flour’, both of which are plausible candidates (both are feminine and highly accessible). Correct disambiguation is achieved through common sense and world knowledge.

In Chinese, zero anaphors take precedence over pronouns at both sentence and discourse levels. This tendency is even more pronounced in recipes: zero anaphora emerges as the predominant means to convey givenness and maintain topic continuity, as exemplified by the numerous instances of zeroes throughout sentences (C.9) to (C.19). Typically, resultant objects are introduced once and subsequently referred to anaphorically through zero anaphors, with their coreferential scope often extending across sentence boundaries. An antecedent might be introduced overtly and then referred to through zeroes, as illustrated by *kuài* ‘pieces (of pumpkin)’ in (C.9) (a), which later serves as the patient in the subsequent predicates *zhēng-shú* ‘steam’, *rù zhàzhījī* ‘insert in juicer’ and *zhà-chéng zhī* ‘juice’. In other cases, the antecedent may be entirely absent, as seen with zero anaphors in sentences (C.12) to (C.15).

- (30) (C.9) (a) 南瓜泥_g, 用南瓜_j去皮切块_k **Ø_k** 蒸熟, (b) 然后 **Ø_k** 入榨汁机 **Ø_k** 榨成汁_l,
 nánguā ní_g yòng nánguā_j qù pí qiē kuài_k **Ø_k** zhēng-shú
 pumpkin puree_g TOP use pumpkin_j remove skin cut pieces_k **Ø_k** steam-cook
 ránhòu_{Ø_k} rù zhàzhījī **Ø_k** zhà-chéng zhī_l
 then **Ø_k** insert juicer **Ø_k** press-become juice_l
 ‘For the pumpkin puree, peel the pumpkin, cut into pieces and steam, then put in the juicer and juice.’

...

- (C.10) 然后除黄油外所有材料_m放入厨师机内。
 ránhòu chú huángyóu wài suǒyǒu cáiliào_m fàng-rù chúshījī nèi
 then except butter out all ingredient put-insert mixer in
 ‘Place all the ingredients in the mixer, excluding the butter.’

- (C.11) 启动搅拌程序然后 **Ø_m** 搅拌 5 分钟。
 qǐdòng jiǎobàn chéngxù ránhòu **Ø_m** jiǎobàn 5 fēnzhōng
 activate mixing program then **Ø_m** mix 5 minutes
 ‘Initiate the mixing program and mix for 5 minutes.’

- (C.12) 加入黄油, 在继续启动搅拌程序, **Ø_n** 搅拌 5 分钟。
 jiārù huángyóu zài jìxù qǐdòng jiǎobàn chéngxù **Ø_n** jiǎobàn 5 fēnzhōng
 add butter again continue activate mixing program **Ø_n** mix 5 minute
 ‘Add the butter and continue mixing for 5 minutes.’

...

- (C.14) **Ø_n** 能拉出膜即可。
Ø_n néng lā-chū mó jíǐě
Ø_n can pull-out film sufficient
 ‘The mixture will be ready when you can pull it out like a film without it breaking.’

(C.15)然后 $\emptyset_n$ 整理成面团 $\circ$, 入不锈钢碗 $_p$ 里。

ránhòu $\emptyset_n$ zhěnglǐ-chéng miàntuán $\circ$ rù bùxiù gāng wǎn $_p$ lǐ
 then $\emptyset_n$ arrange-become dough $\circ$ insert stainless steel bowl $_p$ in
 'Next, knead the mixture until it forms a dough and place it in a stainless-steel bowl.'

....

(C.19) $\emptyset_{p/o}$ 盖上湿布, $\emptyset_o$ 发酵至两倍大, 如图所示。

$\emptyset_{p/o}$ gài-shàng shī-bù $\emptyset_o$ fājiào zhì liǎng-bèi dà rú tú suǒ shì
 $\emptyset_{p/o}$ cover-up wet-cloth $\emptyset_o$ ferment until two-time big as picture NMLZ show
 'Cover it with a damp cloth and allow it to rise until it doubles in volume, as shown in the picture.'

Chinese is commonly acknowledged as a prototypical example of the isolating language type, where verbal morphology cannot contribute disambiguation cues. The role played by inference in anaphora resolution, as expected, is essential. Instances of ambiguous references requiring world knowledge and contextual tracking abound. For example, in (C.15) two full NPs are mentioned: *miàntuán* 'dough' and *bùxiù-gāng-wǎn* 'stainless steel bowl'. Then, five clauses intervene, discussing other matters. In (C.19), a zero occurs as the argument of the verb *fājiào* 'ferment': it is world knowledge that tells us that the antecedent is the dough (which can ferment) and not the bowl.

4.2. Partial coreference and evolutive anaphors

In all three recipes, various instances of partial coreference are present. In English, examples include NPs introduced by numbers, measure words, and adjectives indicating partitivity (e.g., *1/4 inch of water*, *the remaining brown sugar*, *two tablespoons of the remaining butter*) or else with nouns in a whole-part relationship, as e.g., *the insides of the pumpkin*. In Italian, in addition to the means mentioned above (e.g., *un dito d'acqua* 'a finger of water', *parte dell'acqua* 'part of the water'), clitic partitive pronouns also occur, as e.g., 'ne', followed by the quantity--*usate-ne 120gr.* 'use 120 g (of it)' in (I.12). Hypernyms and collective nouns/expressions are used as well, such as *tutto* 'everything' in (I.14) (c) to denote all the aforementioned ingredients.

As expected, several instances of evolutive anaphora are present in all three texts. In the English recipe, the noun *pumpkins* in (E.22) indicates full uncooked pumpkins, while *pumpkin* in (E.29) indicates soft, cooked pumpkin. All mentions in between steps involve some change of state, such as cutting, meat scooping, and baking. Although ontologically different, these evolving entities are encoded by the same full NP, 'pumpkins,' or, alternatively, by pronouns (*them*, *they*) and zero anaphors. The author changes the anaphoric device to 'puree' only in (E.30) when the soft pulp is detached from the rind in (E.29). This can be illustrated through Figure 5.

(31) (E.22) Cut the tops off your **pumpkins**;

(E.23) Cut the **pumpkins** in half, from top to bottom

(E.24) Scoop out the insides of the **pumpkins**

(E.25) Place the **pumpkins** face down on a baking sheet

(E.26) Add about 1/4 inch of water around **them**;

(E.27) Bake $\emptyset_i$ at 350 for 60-90 minutes, checking $\emptyset_j$ periodically after 60 minutes

(E.28) **They** are finished when a fork slides in easily

(E.29) Let the **pumpkin** cool and then scoop the **meat**_k out into a blender.

(E.30) Notes: If you want the **puree**_n to be more like what you get out of the can, drain the **puree** in a colander set over a pot. I found that paper towels worked great for lining the colander.

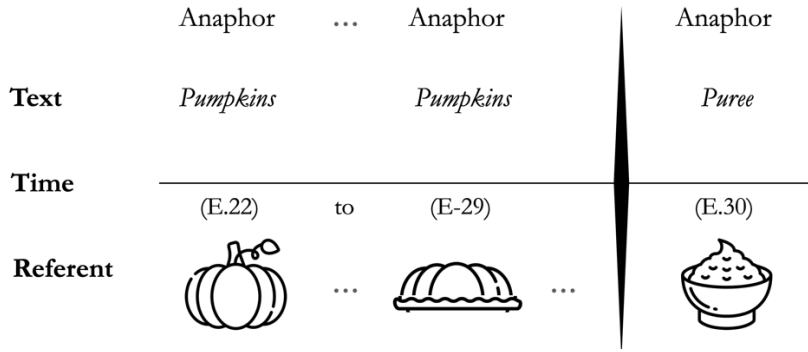


Figure 5: Evolutive anaphors in the English recipe

Another example is provided in sentences (E.76), (E.78), (E.80), and (E.81) above, in which the noun phrase *dough* refers to three distinct entities. In (E.76), it refers to the outcome of the amalgamation of the ingredients, the kneading, and the rising. In (E.78), it represents a flattened dough circle resulting from a ball that constitutes one half of the entire dough quantity. Meanwhile, in (E.80) and (E.81), *dough* refers to a different ball, specifically the other half of the total amount of dough. Evolutive anaphors are encoded through zero anaphors, as well. Similarly, (E.69) to (E.74) below feature zero anaphors that refer to the mix resulting from the combination of the ingredients. However, every sentence refers to a different entity in terms of composition and consistency, as shown in Table 2. In the procedure, the resultant objects of intermediate steps are referred to through zero anaphors: the reader can correctly decode them through inference and world knowledge.

- (32) (E. 69) In a large bowl or the bowl of a stand mixer, dissolve the **yeast in the warm water with a pinch of brown sugar_m**.
 (E.70) Let $\emptyset_m$ sit five minutes, until $\emptyset_m$ foamy.
 (E.71) Add the remaining brown sugar, salt, pumpkin pie spice, egg, egg yolk, pumpkin puree, and 1/4 cup butter_n, and mix $\emptyset_n$ to combine.
 (E.72) Stir in two cups of flour, and mix $\emptyset_n$ to combine.
 (E.73) Add additional flour, 1/4 cup at a time until the **dough_o** comes together and $\emptyset_o$ is only slightly sticky to the touch.

Table 2: evolutive anaphora

Sentence Number	Reference
E.69	<i>yeast in the warm water with a pinch of brown sugar_m</i>
E.70	$\emptyset_m$
E.71	$\emptyset_n$ (= $\emptyset_m$ + flour)
E.72	$\emptyset_o$ (= $\emptyset_n$ + additional flour)
E.73	<i>dough_p</i>

Evolutive anaphors are observable in the Italian and Chinese recipes, as well, with similar referential mechanisms. For Italian, one instance is (I.16)- (I.19) above, where the same referent undergoes several changes of states (in terms of temperature, thickness etc.), while being referred to with the same NP

(*l'impasto*), clitic pronouns, or zero anaphors. In Chinese, evolutive anaphors abound and are mainly encoded through zeroes; moreover, the antecedent may be missing altogether. For instance, zero anaphors in sentences (C.12) to (C.15) refer to the mixture obtained by combining all ingredients ($\mathcal{O}_n$), which is never articulated as a full NP, but instead inferred contextually. Only in (C.15) does the author feel the need to specify that the mixture has now become a *miàntuán* ‘dough’.

5. Discussion and concluding remarks

This paper has delved into the intricate realm of anaphoric phenomena within recipes, exploring not only exact coreference but also partial coreference, bridging, and evolutionary anaphors. The multilanguage approach adopted in this study has shed light on the diverse and nuanced ways in which languages manage anaphoric references in the domain of culinary instructions. While each language exhibits variations in the possibilities for creating anaphoric relations, this genre is generally characterized by a widespread use of zero anaphora. Both English and Italian employ a combination of overt means and zeroes which, crucially, also appear in object position, a phenomenon seldom encountered in other linguistic genres. Italian, however, also encodes highly activated objects with clitic pronouns that agree in gender and number with nouns and verbal suffixes, thereby allowing easier coreference disambiguation. Conversely, Chinese exhibits greater reliance on zero anaphora wherein, in certain instances, overt antecedents are absent. Despite this, antecedent retrieval is allowed thanks to inferential processes/world knowledge. Importantly, inference proves indispensable not only in inference-based tracking languages like Chinese but also in Italian and English, confirming its crucial role in anaphora resolution across languages. Any AI systems aiming to effectively address anaphora resolution and achieve complete textual comprehension must integrate inferential processes at a certain stage in their design and implementation.

Moreover, the study has also uncovered instances where anaphoric means are selected irrespective of the activation status of antecedents, contrary to what is predicted by the theories outlined in Section 2.2. Notably, full NP repetition is observed also within the same sentence or step, i.e., with highly activated referents. Additionally, zero anaphora is identified in cases where the referent's last overt mention is relatively distant, hence, relatively inactive. This latter phenomenon is not exclusive to Chinese, where such occurrences are common, particularly in colloquial registers. Surprisingly, it also manifests in English, a language that is generally less tolerant in this respect. These findings challenge established expectations and highlight the nuanced and varied ways in which anaphoric reference operates in this genre. The use of both repetition and zero anaphora can be attributed to the dual nature of recipe texts. On one hand, they exhibit characteristics typical of procedural discourse, favouring precise reference encoding to avoid ambiguity. On the other, their colloquial nature, enriched with blog-like register colloquialisms, renders them conducive to the use of synonyms, hypernyms, pronouns, and a general avoidance of repetition. This duality reflects the unique linguistic demands posed by recipes, which blend the need for clarity in conveying step-by-step instructions with the informality and flexibility inherent in everyday language use. Evolutive anaphora, finally, is undeniably a widespread phenomenon in recipes across various languages. Grasping the nuances of evolutive anaphora requires a keen understanding of how references evolve over the course of instructions, adding an additional layer of intricacy to the already complex landscape of linguistic phenomena within culinary texts.

Notes: For academic purposes, Anna Morbiato is responsible for Sections 1, 2, 3, 4, 4.2, and 5; Anna Morbiato and Patrizia Cani are jointly responsible for Sections 4.1. In this paper, we use the term ‘Chinese’ to refer to Pǔtōnghuà, the standard language of the PRC. Throughout the paper, glosses follow the general guidelines of the Leipzig Glossing Rules. Additional glosses: COS = ‘change of state’; PM = ‘patient marker’. List of abbreviations: C. = Chinese; E. = English; I. = Italian.

6. References

- Asher, Nicholas. 2000. "Events, Facts, Propositions, and Evolutive Anaphora." In *Speaking of Events*, edited by Achille Varzi, James Higginbotham, and Fabio Pianesi, 123–50. Oxford: Oxford University Press.
- Asher, Nicholas, and Alex Lascarides. 1998. "Bridging." *Journal Of Semantics* 15 (1): 83–113.
- Bender, Emily. 1999. "Constituting Context: Null Objects in English Recipes Revisited." *Penn Working Papers in Linguistics* 6 (1): 53–68.
- Bickel, Balthasar. 2011. "Grammatical Relations Typology." In *The Oxford Handbook of Linguistic Typology*, edited by Jae Jung Song, 399–444. Oxford: Oxford University Press.
- Bies, Ann, Mark Ferguson, Karen Katz, and Robert Mac-Intyre. 1995. "Bracketing Guidelines for Treebank II Style."
- Bois, John W Du. 1980. "Beyond Definiteness: The Trace of Identity in Discourse." *The Pear Stories: Cognitive, Cultural, and Linguistic Aspects of Narrative Production* 3: 203–74.
- Bosselut, Antoine, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. "Simulating Action Dynamics with Neural Process Networks." *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 1–15.
- Brown, Gillian, and George Yule. 1983. *Discourse Analysis*. Cambridge: Cambridge University Press.
- Chafe, Wallace L. 1976. "Givenness, Contrastiveness, Definiteness, Subjects, Topics, and Points of View." In *Subject and Topic*, edited by Charles Li, 25–55. New York: Academic Press.
- . 1994. *Discourse, Consciousness, and Time: The Flow and Displacement of Conscious Experience in Speaking and Writing*. Chicago and London: The University of Chicago Press.
- Chen, Chen, and Vincent Ng. 2016. "Chinese Zero Pronoun Resolution with Deep Neural Networks." *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers* 2: 778–88.
- Chung, Tagyoung, and Daniel Gildea. 2010. "Effects of Empty Categories on Machine Translation." *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, no. October: 636–45.
- Clark. 1977. "Bridging." In *Thinking: Readings in Cognitive Science.*, edited by Philip Nicholas Johnson-Laird and Peter Cathcart Wason, 411–20. Cambridge: Cambridge University Press.
- Clark, H H, and S E Haviland. 1975. *Comprehension and the Given-New Contract*. Social Sciences Working Paper. School of Social Sciences, University of California, Irvine.
- Comrie, Bernard. 1989. *Some General Properties of Reference-Tracking Systems*. na.
- Delmonte, Rodolfo. 2009. *Computational Linguistic Text Processing – Lexicon, Grammar, Parsing and Anaphora Resolution*. New York: Nova Science Publ.
- . 2014. "Predicate Argument Structures for Information Extraction from Dependency Representations: Null Elements Are Missing." In *Distributed Systems and Applications of Information*

Filtering and Retrieval, Studies in Computational Intelligence, edited by C. Lai. Verlag Berlin Heidelberg:

- Dijk, Teun A. van, and Walter Kintsch. 1983. *Strategies of Discourse Comprehension*. New York: Academic
- Fang, Biaoyan, Timothy Baldwin, and Karin Verspoor. 2022. “What Does It Take to Bake a Cake? The RecipeRef Corpus and Anaphora Resolution in Procedural Text.” *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 3481–95.
- Fang, Biaoyan, Christian Druckenbrodt, Saber A Akhondi, Camilo Thorne, Timothy Baldwin, and Karin Verspoor. 2022. “Annotation Guideline for RecipeRef : A Corpus for Modeling Anaphora Resolution of the Procedural Text of Recipes.”
- Favrel, José, and Pierre Barrouillet. 2000. “On the Relation between Representations Constructed from Text Comprehension and Transitive Inference Production.” *Journal of Experimental Psychology: Learning, Memory, and Cognition*. US: American Psychological Association.
- Foley, William A, and Robert D. Van Valin. 1984. *Functional Syntax and Universal Grammar*. Cambridge: Cambridge University Press.
- Garrod, S, D Freudenthal, and E Boyle. 1994. “The Role of Different Types of Anaphor in the On-Line Resolution of Sentences in a Discourse.” *Journal of Memory and Language* 33 (1): 39–68.
- Givón, Talmy. 1983. *Topic Continuity in Discourse: A Quantitative Cross-Language Study*. John Benjamins Publishing.
- Gordon, Peter C., Barbara J. Grosz, and Laura A. Gilliom. 1993. “Pronouns, Names, and the Centering of Attention in Discourse.” *Cognitive Science* 17 (3): 311–47.
- Greene, Steven B., Richard J. Gerrig, Gail McKoon, and Roger Ratcliff. 1994. “Unheralded Pronouns and Management by Common Ground.” *Journal of Memory and Language*.
- Halliday, Michael Alexander Kirkwood, and Ruqaiya Hasan. 1976. *Cohesion in English*. London: Taylor & Francis.
- Huang, Yan. 2000. *Anaphora: A Cross-Linguistic Study*. Vol. 3. New York: Oxford University Press.
- Iida, Ryu, and Massimo Poesio. 2011. “A Cross-Lingual {ILP} Solution to Zero Anaphora Resolution.” In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 804–13. Portland, Oregon, USA: Association for Computational Linguistics.
- Iwata, Sei, Taro Watanabe, and Masaaki Nagata. 2021. “Zero Pronouns Identification Based on Span Prediction.” *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Student Research Workshop*, no. 2020: 331–36.
- Jackendoff, Ray S. 1990. *Semantic Structures*. Cambridge: MIT Press.
- Jiang, Yiwei, Klim Zaporozets, Johannes Deleu, Thomas Demeester, and Chris Develder. 2020. “Recipe Instruction Semantics Corpus (RISeC): Resolving Semantic Structure and Zero Anaphora in Recipes.” *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the*

10th International Joint Conference on Natural Language Processing, no. Section 3: 821–26.

- Kim, Youngtae, Dongyul Ra, and Soojong Lim. 2021. “Zero-Anaphora Resolution in Korean Based on Deep Language Representation Model: BERT.” *ETRI Journal* 43 (2): 299–312.
- Kintsch, Walter. 1988. “The Role of Knowledge in Discourse Comprehension: A Construction-Integration Model.” *Psychological Review*. US: American Psychological Association.
- Kintsch, Walter, and Teun A van Dijk. 1978. “Toward a Model of Text Comprehension and Production.” *Psychological Review*. US: American Psychological Association.
- Korzen, Iørn. 2009. “Tipologia Anaforica: Il Caso Della Cosiddetta ‘Anafora Evolutiva.’” Edited by Accademia della Crusca. *Studi Di Grammatica Italiana* XXV: 323–57.
- LaPolla, Randy J. 1993. “Arguments Against ‘Subject’ and ‘Direct Object’ as Viable Concepts in Chinese.” *Bulletin of the Institute of History and Philology* 63 (4): 759–813.
- Lata, Kusum, Pardeep Singh, and Kamlesh Dutta. 2021. *A Comprehensive Review on Feature Set Used for Anaphora Resolution*. *Artificial Intelligence Review*. Vol. 54. Springer Netherlands.
- Levinson, Stephen C. 1987. “Pragmatics and the Grammar of Anaphora: A Partial Pragmatic Reduction of Binding and Control Phenomena.” *Journal of Linguistics* 23 (2): 379–434.
- . 2000. *Presumptive Meanings: The Theory of Generalized Conversational Implicature*. MIT Press.
- Li, Wendan. 2005. *Topic Chains in Chinese: A Discourse Analysis and Application in Language Teaching*. München: Lincom Europa.
- Lucas, Margery M, Michael K Tanenhaus, and Greg N Carlson. 1990. “Levels of Representation in the Interpretation of Anaphoric Reference and Instrument Inference.” *Memory & Cognition*. US: Psychonomic Society.
- Mandler, Jean Matter. 1984. *Stories, Scripts, and Scenes*. New York and London: Psychology Press.
- Massam, Diane, and Yves Roberge. 1989. “Recipe Context Null Objects in English.” *Linguistic Inquiry* 20 (1): 134–39.
- Mitkov, Ruslan. 1999. “Anaphora Resolution - The State of the Art.”
- . 2022. “Anaphora Resolution.” In *The Oxford Handbook of Computational Linguistics*, edited by Ruslan Mitkov. Oxford and New York: Oxford University Press.
- Morbiato, Anna. 2015. “Quello Che i Cinesi Non Dicono.” In *Lingua Cinese: Variazioni Sul Tema (Sinica Venetiana 2)*, edited by Magda Abbiati and Federico Greselin, 79–102. Edizioni Ca’ Foscari.
- . 2022. “Pragmatics of Narration with Language.” In *Foundations for Meaning and Understanding in Human-Centric AI*, edited by Luc Steels, 93–106. Venice: Venice International University.
- Norrick, Neal R. 1983. “Recipes as Texts: Technical Language in the Kitchen. In , 73-181. Berl.” In *Sprache, Diskurs Und Text*, edited by Rene Jongen, Sabine De Knop, Peter H Nelde, and Marie-Paule Quix,

73–181. Berlin, Boston: De Gruyter.

- Oguz, Cennet, Ivana Kruijff-Korabayova, Emmanuel Vincent, Pascal Denis, and Josef Van Genabith. 2022. “Chop and Change: Anaphora Resolution in Instructional Cooking Videos.” *2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing - Findings of the Association for Computational Linguistics: ACL-IJCNLP 2022*, 364–74.
- Song, Linfeng, Kun Xu, Yue Zhang, Jianshu Chen, and Dong Yu. 2020. “ZPR2: Joint Zero Pronoun Recovery and Resolution Using Multi-Task Learning and BERT.” *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 5429–34.
- Tao, Hongyin. 2007. “Cāozuò Yǔ Tǐ Zhōng Dòngcí Lùn Yuán Jiégòu de Shíxiàn Jí Yǔ Yòng Yuánzé 操作语体中动词论元结构的实现及语用原则 [The Discourse Pragmatics of Argument Structure in Chinese Procedural Discourse].” *Zhongguo Yuyan* 1: 3–13.
- Tao, Liang. 1996. “Topic Discontinuity and Zero Anaphora in Chinese Discourse.” In *Studies in Anaphora*, edited by Barbara A. Fox, 487–513. Amsterdam, Philadelphia: John Benjamins Publishing.
- . 2001. “Switch Reference and Zero Anaphora: Emergent Reference in Discourse Processing.” In *Conceptual and Discourse Factors in Linguistic Structure*, edited by Alan Cienki, Barbara J. Luka, and Michael B. Smith, 253–69. Stanford: CSLI Publications.
- Tao, Liang, and Alice F. Healy. 2005. “Zero Anaphora: Transfer of Reference Tracking Strategies from Chinese to English.” *Journal of Psycholinguistic Research* 34 (2): 99–131.
- Vonk, Wietske. 1985. “The Immediacy of Inferences in the Understanding of Pronouns.” In *Inferences in Text Processing*, edited by Gert Rickheit and Hans B. T. - Advances in Psychology Strohner, 29:205–18. North-Holland.
- Wharton, Tim. 2010. “Recipes: Beyond the Words.” *Gastronomica* 10 (4): 67–73.
- Xue, Nianwen, and Fei Xia. 2000. “2000. The Bracketing Guide- Lines for the Penn Chinese Treebank. Technical Report IRCS-00-08.”
- Yamakata, Yoko, Shinsuke Mori, and John Carroll. 2020. “English Recipe Flow Graph Corpus.” *LREC 2020 - 12th International Conference on Language Resources and Evaluation, Conference Proceedings*, no. May: 5187–94.
- Yin, Qingyu, Yu Zhang, Weinan Zhang, Ting Liu, and William Yang Wang. 2018. “Deep Reinforcement Learning for Chinese Zero Pronoun Resolution.” *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)* 1: 569–78.

Chapter 8

Can You Pass the Salt? An Ontological Representation of Imperative Meaning

Vanja Cangalovic

Robert Porzel

Mihai Pomarlan

Bremen University, Bremen, Germany

Abstract

There is a distinct difference between modeling linguistic knowledge and knowledge that is expressed linguistically. The research effort described herein concerns the latter, i.e. how to model the output of a natural language understanding system in a formal ontology in such a way that robotic agents can carry out the tasks given to them via natural language. For this, we build on a given foundational ontology that is suitable for our requirements and introduce the basic modeling principles and design patterns to model and represent the meaning of instructions. Linguistically, our model is informed by constructional approaches to language understanding, i.e. embodied- and fluid construction grammar. Ontologically, we base our model on the foundational framework provided by the Dolce Ultra Light + Descriptions and Situations ontology. The proposed model, called SOMA-SAY, is part of a larger system of ontologies that are employed in the Everyday Activity Science and Engineering collaborative research center.

Keywords: Natural Language Understanding, Everyday Activities, Imperatives

8.1 Introduction

Numerous formal ontological models representing linguistic objects and their corresponding grammatical theories have been presented in the past. This is not another one of these models. While we will discuss some prior modeling efforts in Section 8.2, the scope and intent of our model differs from previous ones in several respects:

- This work does not focus on modeling linguistic objects themselves, such as morphological, lexical or syntactic constructions.
- This work does focus on representing the socially constructed meaning that is conveyed by instructions,

- Specifically, our work takes place in the challenging and practical application of giving robots underspecified and vague natural language instructions to perform certain everyday activities that are subsequently carried out by them. Representing an underspecified instruction also involves representing information that was left implicit by the textual or verbal command.

The ontology module presented in this work is part of the cognitive backbone of a greater robotic infrastructure [1, 3]. It serves the function of representing the socially constructed semantic content of an instruction. More specifically, given that we are using a constructional approach to language understanding [33], we need to represent the pragmatic implications of the meaning pole of imperative constructions. This work, therefore, constitutes an attempt to employ state of the art knowledge engineering principles to connect formal ontologies to the cognitive linguistic theories of the so-called *West-coast* school of linguistics [21, 19, 22, 9].

8.2 State of the Art

In this section we present related work on modeling linguistic knowledge as well as the foundational and ontological backdrop on which this work resides. Additionally, we will briefly highlight the natural language understanding approach that yields the corresponding structures described herein.

8.2.1 Modeling Linguistic Knowledge

In the past 20 years various approaches to model linguistic knowledge, i.e. the entities and features that make up human language, in formal ontologies have been proposed. These approaches differ in some respects such as alignment to upper layers, their modeling intent and their scope. One point of divergence lies in the specific alignment to a foundational layer. While, for example, the GOLD ontology [7] is aligned to the SUMO upper ontology [21], the OntoWordNet model [10], is aligned to the DOLCE foundational ontology [16]. The LingInfo model [3] can be used with any foundational framework as it relies on meta-classes to model information about the lexical entities. As for representing pragmatically relevant information, the model proposed herein can be seen as an extension of the models proposed by Loos and Porzel [23].

In contrast, the OntoWordNet aims at merging the linguistic information contained in WordNet with the respective classes employed in specific domain models, while both LingInfo and GOLD seek to incorporate more linguistic information, such as morphological and grammatical features of language. They all allow for a direct connection of the respective linguistic information for terms with corresponding classes and properties in a domain ontology.

These efforts are, in a sense, orthogonal to ours and each model could be integrated as an additional module to allow reasoning about linguistic information or as a link between lexical and ontological resources. For those purposes we actually employ a *Lower Semantic Model* that connects lexical and ontological information and is interchangeable with either of the models described above. A little closer to home is the so-called *General Upper Model* (GUM) [2].

GUM provides a detailed semantics for linguistic spatial expressions and is based on a principled ontological engineering approach. It covers language concerned with space, actions in space and spatial relationships for which an ontological organization is proposed that relates such expressions to general classes of fixed semantic import. However, as we seek to align our model with a specific foundational model and construct it as a module within the SOMA ontological

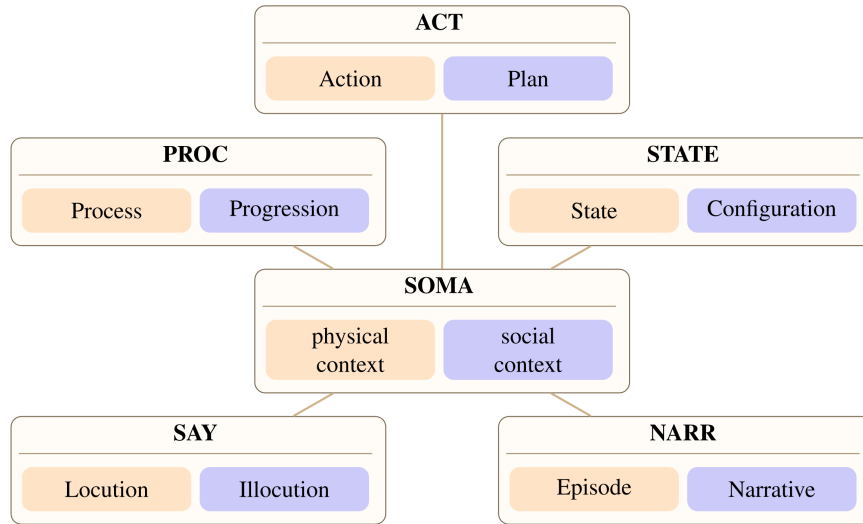


Figure 8.1: Part of the modular organization of SOMA and its SAY module. Each module has a grounded and descriptive part..

framework [7], discussed below, we cannot employ the upper model GUM as is, but will re-use relevant details concerning schematic theories about functional relations, where applicable.

8.2.2 The EASE Ontology: SOMA

The ontology module described in this work is part of the EASE family of ontologies called SOMA¹ as depicted in Figure 8.1. As SOMA it is, therefore, based on the DOLCE+DnS Ultra-lite (DUL) foundational framework [26]. This decision is greatly motivated by their underlying ontological commitments. Firstly, DUL is not a revisionary model, but seeks to express stands that shape human cognition. Furthermore, it assumes a reductionist approach – rather than capturing, for example, the flexibility of our usage of objects via multiple inheritance in a multiplicative manner, it commits to a reduced *ground* classification and use a *descriptive* approach for handling this flexibility. For this, a primary branch of the ontology represents the ground physical model, e.g. objects and actions, while a secondary branch represents the social model, e.g. roles and tasks. All entities in the social branch would not exist without humans, i.e. they constitute social objects that represent concepts about or descriptions of ground elements. Every axiomatization in the physical branch can, therefore, be regarded as expressing some physical context whereas axiomatizations in the descriptive social branch are used to express social contexts. A set of dedicated relations is provided that connect both branches. For example the relation *classifies* connects ground objects, e.g. specific utterances, with the roles they can play, i.e. potential classifications. Thus, we can state that an interrogative can in some context be conceptualized as an assertion, a command or simply a query. Nevertheless, neither its ground ontological classification as an interrogative will change nor will interrogatives be subsumed as kinds of commands or assertions via multiple inheritance.

The semantics of the actions and entities executed by the robots performing everyday activities, such as setting a table or cooking a meal, are defined by this formal ontology. This ontology is designed to provide descriptions for everyday activities in terms of human physiology and human mental concepts, as well as enabling formal reasoning. The ontology supplying the labels for the objects has been designed using the principles proposed by Masolo et al. [16, 24]. Specific branches of the former KnowRob knowledge model pertaining to everyday activities

¹SOMA stands for *Socio-physical Model of Activities* and is described in detail in Bessler et. al [7].

[3], such as those involved in table setting and cooking, have consequently been aligned to the DUL framework in the creation of the SOMA framework. Additional axiomatization that is beyond the scope of description logics is integrated by means of the Distributed Ontology Language [27].

Generally speaking, the DUL foundational ontology constitutes an axiomatic theory about the high-level domain-independent categories in the real world [16, 12], such as object, attribute, event, spatial and temporal connections and the like. The purpose of this foundational ontology is to act as a modeling basis for building individual domain ontologies, e.g., an ontology of actions (ACT) or one for objects (OBJ) as shown in Figure 8.1. Equally important is that foundational ontologies provide ontology design patterns that prescribe best practice modeling, avoid ontological idiosyncrasies, and save a substantial amount of modeling effort [9].

In order to highlight an ontology design pattern pertinent to the work discussed herein, we will take a look at the ACTION versus TASK distinction. Any ACTION is an EVENT with at least one AGENT that *isParticipantIn* it, and that executes a TASK that *isDefinedIn* a PLAN, WORKFLOW or PROJECT. An ACTION is that what actually happens in the world and a TASK is a CONCEPT that humans employ. If a person lets go of an object and it ends up on the table this could be the execution of a PLACINGTASK or a THROWINGTASK depending on the given conceptualization thereof.

8.2.3 Natural Language Understanding in EASE

Our application domain requires robotic agents to carry out natural language instructions, such as recipes or manuals. This involves flexibly parsing abstract and underspecified input into a meaning representation, which in turn enables further steps necessary for real grounding, such as simulations or additional reasoning. To this end, a deep semantic parser based on the Construction Grammar formalism is employed [10, 15, 11]. Both the constructions' meaning poles and the analysis itself make use of ontological knowledge, to disambiguate otherwise unclear instructions and to evoke unspecified parameters which need to be inferred by later processing steps. In this way, natural language commands are transformed into models of SOMA-SAY, which consist of series of state transitions specifying the evoked schemas.

The employed parser's underlying mechanism is based on the unification and merging algorithms implemented in Fluid Construction Grammar[34], while its ontology integration and schema handling are inspired by the Embodied Construction Grammar framework [5]. The grammar itself is hand-engineered for the domain of household instructions. The parsing process integrates tightly into our knowledge base-focused understanding pipeline, with both the internal data structures as well as the semantic output being represented as semantic triples.

8.3 The SOMA-SAY Module

As described in Section 8.2.2 we base our model on the DUL+D&S foundational ontology as a module of the larger SOMA ontology. This enables us to firstly define a LOCUTION as an EVENT of type ACTION (8.1).

$$\begin{aligned}
\text{Locution} &\sqsubseteq \text{Action} & (8.1) \\
\text{LinguisticObject} &\sqsubseteq \text{InformationObject} & (8.2) \\
\text{Declarative} &\sqsubseteq \text{LinguisticObject} & (8.3) \\
\text{Instructive} &\sqsubseteq \text{LinguisticObject} & (8.4) \\
\text{Interrogative} &\sqsubseteq \text{LinguisticObject} & (8.5) \\
\text{Locution} &\sqsubseteq \forall \text{hasPart.LinguisticObject} & (8.6)
\end{aligned}$$

A LOCUTION represents any activity of expression using natural language that can be verbal, textual or else. The actual linguistic form of such an expression, i.e. declaratives, imperatives or interrogatives, are modeled as a LINGUISTICOBJECT (8.3-8.5) which itself is an INFORMATIONOBJECT (8.2). Using the given *hasPart* relation we can express that only linguistic objects can be components of a locution (8.6).

The DOLCE+DnS Ultralite foundational ontology features the useful distinction between an EVENT that happens in the world and an EVENTTYPE, i.e. a classification or interpretation thereof.² Analogously, we can apply the same design pattern to the linguistic function of a LOCUTION, i.e. an ILLOCUTION (8.7,8.11). Additionally, we define three types of ILLOCUTION, i.e. ASSERTION, INSTRUCTIVE and INTERROGATIVE (8.8-8.10).

$$\begin{aligned}
\text{Illocution} &\sqsubseteq \text{EventType} & (8.7) \\
\text{Assertion} &\sqsubseteq \text{Illocution} & (8.8) \\
\text{Command} &\sqsubseteq \text{Illocution} & (8.9) \\
\text{Query} &\sqsubseteq \text{Illocution} & (8.10) \\
\text{Illocution} &\sqsubseteq \forall \text{classifies.Locution} & (8.11)
\end{aligned}$$

This solves certain aspects of semantic construal and pragmatic analysis [20, 32]. We all know that utterances formalized as declaratives, e.g. *it is drafty in here*, are actually veiled instructions as in this example to close a window or door. Our models allows for all possible construals, i.e. questions that are meant as assertions, e.g. *how stupid is this* meaning *this is very stupid* as well as all other form-function combinations. This is realized via the *classifies* relation shown in Axiom 8.11.

In our application domain instructions are of particular importance, as they constitute the input for the robotic agents. So even if asked *could you put the plates on the table* the robot should in most cases not reply with a resounding *yes*, but rather execute the given command. In the Descriptions & Situation system [9] a DESCRIPTION is a SOCIALOBJECT that represents a conceptualization that can be understood as a 'descriptive context'. This context uses or defines concepts in order to create a view on a 'relational context'. This relational context is modeled as a SITUATION.

A SITUATION is a view consistent with a DESCRIPTION, i.e. it *satisfies* it. It is created by an observer on the basis of a 'frame' provided by the DESCRIPTION. As such it used to represent reified n-ary relations, where *isSettingFor* is the top-level relation for all binary projections of the n-ary relation. For modeling the meaning of any locution that is construed as a command we

²DUL defines an EVENTTYPE as a concept that *classifies* an EVENT. It should, therefore, describe how an EVENT should be interpreted, executed, expected, seen, etc.

employ the `TRANSITION` subclass of `SITUATION`. First of all a `TRANSITION` creates a context for two additional different `Situation(s)`, i.e. the state before and the state after the transition takes place. For our scenario we create a subclass of `TRANSITION` called `STATETRANSITION` (8.12). Via the relation *includesEvent*, we can state that every `STATETRANSITION` includes exactly two `EVENTS` (8.13).

$$\text{StateTransition} \sqsubseteq \text{Transition} \quad (8.12)$$

$$\text{StateTransition} \sqsubseteq = 2.\text{includesEvent.Event} \quad (8.13)$$

Additionally, we create a new type of situation, called a `SCENE` (14). Using the given DUL relations of *hasInitialState* and *hasTerminalState*, we can connect situations to situations, i.e. two scenes to one transition, as shown in Axioms 8.15 and 8.16.

$$\text{Scene} \sqsubseteq \text{Situation} \quad (8.14)$$

$$\text{StateTransition} \sqsubseteq = 1.\text{hasInitialState.Scene} \quad (8.15)$$

$$\text{StateTransition} \sqsubseteq = 1.\text{hasTerminalState.Scene} \quad (8.16)$$

Based on the constructional approach to language understanding, described in Section 8.2.3, representing the meaning of utterances is impossible without recourse to schematic knowledge in the form of image schema, x-schema or c-schema as modeled in FrameNet [21, 19, 22, 9, 1]. We, therefore, subsume these schematic theories under the class `THEORY` making them a type of `DESCRIPTION` (8.17-8.19).

$$\text{SchematicTheory} \sqsubseteq \text{Theory} \quad (8.17)$$

$$\text{ImageSchemaTheory} \sqsubseteq \text{SchematicTheory} \quad (8.18)$$

$$\text{ExecutableSchemaTheory} \sqsubseteq \text{SchematicTheory} \quad (8.19)$$

We can, now, start to populate our model with the schemas that have been proposed in the literature [21, 19] with the addition of an `ESTSCHEMA` that is evoked in expressions featuring the existence of an entity, as in "there is a cup" (8.20-8.22).

$$\text{SourcePathGoalSchema} \sqsubseteq \text{ImageSchemaTheory} \quad (8.20)$$

$$\text{TrajectorLandmarkSchema} \sqsubseteq \text{ImageSchemaTheory} \quad (8.21)$$

$$\text{EstSchema} \sqsubseteq \text{ImageSchemaTheory} \quad (8.22)$$

We now define a new *evokes* relation that can hold between two social objects (8.23). In our case we employ it to state that schemas can evoke other schemas (8.24), thereby making the evoked schemas' constituent roles accessible to itself [5].

$$\text{SocialObject} \sqsubseteq \forall \text{evokes.SocialObject} \quad (8.23)$$

$$\text{ContainerSchema} \sqsubseteq \exists \text{evokes.LocationSchema} \quad (8.24)$$

Following the practice of using these schema computationally [28, 5], we want to assign constituents to the individual schemas, whereby only entities of type `ROLE` can be a constituent of a schema (8.25-8.29).

$$\text{SchematicTheory} \sqsubseteq \forall \text{hasConstituent.Role} \quad (8.25)$$

$$\text{SourcePathGoalSchema} \sqsubseteq \exists \text{hasConstituent.Source} \quad (8.26)$$

$$\text{SourcePathGoalSchema} \sqsubseteq \exists \text{hasConstituent.Path} \quad (8.27)$$

$$\text{SourcePathGoalSchema} \sqsubseteq \exists \text{hasConstituent.Goal} \quad (8.28)$$

$$\text{SourcePathGoalSchema} \sqsubseteq \exists \text{hasConstituent.Trajector} \quad (8.29)$$

As provided by the DUL foundations the classification of objects is realized via the `ROLE` pattern provided by the foundational layer. Roles are `CONCEPTS` and, as such, reside in the `SOCIALOBJECT` branch of DUL. The *classifies* relation is used to constrain what can be classified as a source or goal or what can be classified as a trajector, as exemplified in Axioms 8.30 through 8.35. This role pattern is of paramount importance, especially in restricting the type of entity that become the filler of a schematic constituent. The SOMA framework imports the roles that have been established in the field of frame semantics [1]. The selectional restrictions imposed by the *classifies* relation are used in a number of reasoning processes ranging from natural language understanding to tool selection. As certain roles can only classify physical agents or specific types of designed artifacts these axiomatizations provide substantial information about context dependent *meaning* of objects.

$$\text{Goal} \sqsubseteq \text{Role} \quad (8.30)$$

$$\text{Goal} \sqsubseteq \forall \text{classifies.Location} \quad (8.31)$$

$$\text{Source} \sqsubseteq \text{Role} \quad (8.32)$$

$$\text{Source} \sqsubseteq \forall \text{classifies.Location} \quad (8.33)$$

$$\text{Trajector} \sqsubseteq \text{Role} \quad (8.34)$$

$$\text{Trajector} \sqsubseteq \forall \text{classifies.Entity} \quad (8.35)$$

Lastly, we can now link situations to schematic theories, as every scene and action in a given situation evokes specific schematic theories that consequently need to be satisfied.

$$\text{StateTransition} \sqsubseteq \forall \text{satisfies.SchematicTheory} \quad (8.36)$$

$$\text{Scene} \sqsubseteq \forall \text{satisfies.ImageSchemaTheory} \quad (8.37)$$

$$\text{Action} \sqsubseteq \forall \text{satisfies.ExecutableSchemaTheory} \quad (8.38)$$

In the following section, we will exemplify how this model is used to express the output of constructional semantic parsing in our system. Please note that the ensuing structures are still not sufficiently specific to drive a robotic control system. A detailed overview of further parametrizations and explication of these plans via physical simulation and human computation is given in Pfau et al. [30].

8.4 The Model at Work

The SOMA-SAY module is the ontological foundation of our natural language understanding process in several ways:

- It constitutes the meaning pole of the constructions contained in the grammar.
- It is actively employed in the parsing process.
- It defines the target representation of the parsing output.

In the following we will exemplify the parsing process and its resulting output, i.e. semantic specifications, as well as our solution for binding role fillers of evoked schemas to each other.

8.4.1 Creating Semantic Specifications

To provide a picture of the parsing process, we take a look at an example instruction. This example is selected as it shows how the semantic output is aligned with the SOMA-SAY module. Furthermore, this example also highlights the benefits of employing ontological knowledge during the parsing process.

- (1) Crack the eggs into the pan.
- (2) Cover it.

The semantic parsing result for the instructions, given in Example 1 and Example 2, is shown in Figure 8.2. For readability purposes, the output is shortened to contain only directly relevant, semantic triples.

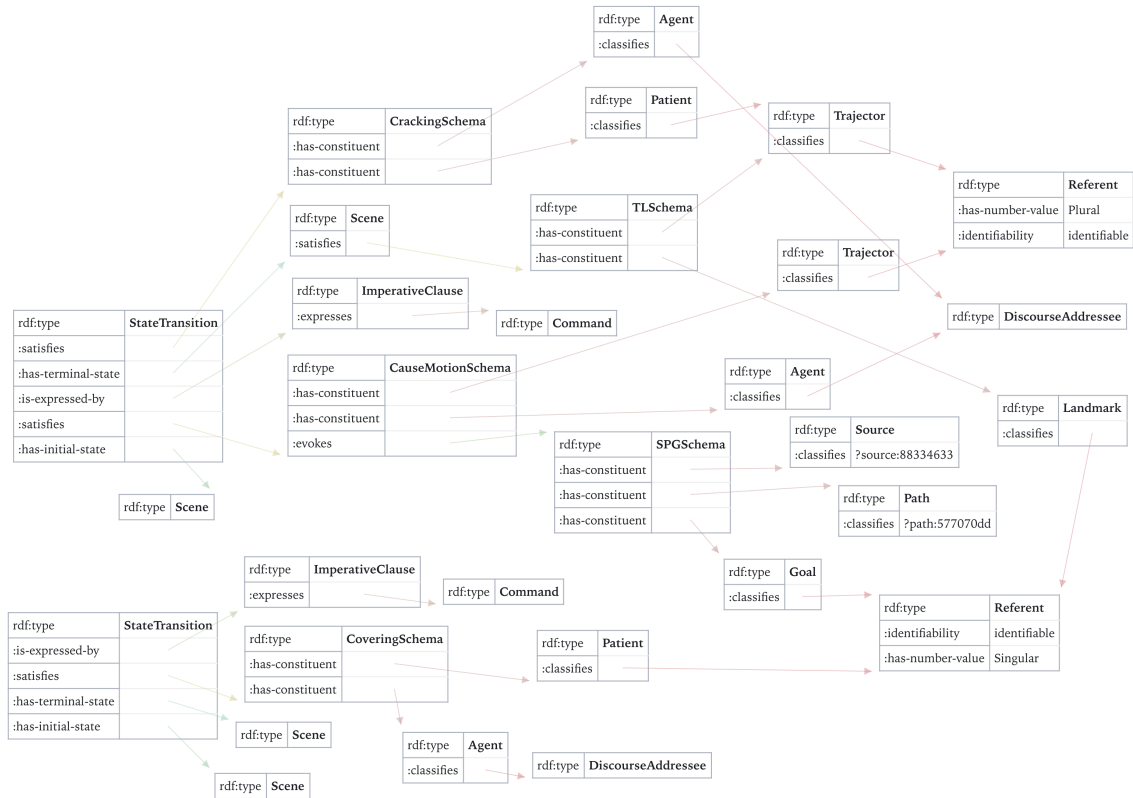


Figure 8.2: This image shows the semantic output of the parser given Example ?? and ?? as input.

The sequence of interrogative clauses is parsed into two COMMANDS, whose actions are to be performed by the discourse addressee. For the cracking task, the patient "eggs" is identified, while the obligatory constituent role of INSTRUMENT is left unspecified. Moreover, the evoked CAUSED-MOTION-SCHEMA interprets the first clause as referring to an action in which the agent causes the patient to move along some path. Thus, the semantic output specifies the action to take place on a PATH from an unspecified SOURCE location towards the "pan", while the terminal scene further specifies the type of the final spatial placement by satisfying the CONTAINER-SCHEMA.

The subsequent clause, given in Example ??, directly evokes the COVERING-SCHEMA, whose agent is the same discourse addressee, whereas the patient is identified as "it". This personal pronoun refers to the expression "pan" from the preceding action. A simple coreference resolution process looking for the nearest preceding referring expression with the same ontological category makes the coreference explicit by binding the respective referent individuals. The CAUSED-MOTION-SCHEMA's goal and COVERING-SCHEMA's patient roles thus classify the same filler instance.

Additionally, each unnegated, nonhypothetical mention of an entity is taken to assert its own existence in the respective clause's scenes, which is represented by each respectively evoked EST-SCHEMA. However, to maintain readability, these schemata are not illustrated in Figure 8.2.

A benefit of directly employing ontological knowledge during the parsing process is the potential for disambiguation. For instance, the instruction given in Example ?? is syntactically very similar to instruction of Example ??, however the conveyed meanings clearly differ.

(3) Knead the dough into a ball.

In Example 3, instead of describing an action to be carried out on a path, the manipulating action is required to result in the patient having a specific configuration. Thus, when parsing instruction ??, both the CAUSED-MOTION-CXN and RESULTATIVE-CXN compete for application, the only difference being their expected ontological category for the final noun phrase's referent. As "ball" is a type of geometric form, it satisfies the filler constraint of the RESULTATIVE-CXN, which is able to apply and contribute its meaning.

The linguistic constructions constituting the instructions in Examples ??, ?? and ?? directly evoke several schemas, as shown in Figure 8.2. Some of these, for instance the CAUSED-MOTION-SCHEMA, in turn evoke additional schemas, which are used as a setting for their respective definitions and allow for the binding of equivalent role fillers as described in Section 8.4.2 below.

8.4.2 Binding Fillers

When dealing with feature structures one needs to represent both information about which entities fill in the role of the feature structure, as well as constraints on fillers of different roles, possibly from different feature structures, such as ones imposing identity between two fillers.

Representing such constraints at the T-box level in OWL-DL is impossible because of global restrictions to ensure decidability. This means, we cannot impose e.g. that a pair of schemas of a particular type must always share role fillers. Nevertheless, we want to explicitly flag such constraints when they appear for instances of schemas, for use with reasoners outside OWL.

To this end, we define a BINDING as a reified relation holding between a Role or Parameter and its filler entity, where this relation only holds within some descriptive context, such as

the workflow of a task. Bindings would then be asserted as individuals in an A-box, i.e. a knowledge graph, representing some collection of schemas.

Naturally, the question arises of why not to just use the *classifies* property provided by DUL for this purpose. The reason is that Bindings allow for more sophisticated connections such as `ROLEROLEBINDINGS`, `CONDITIONALBINDINGS` and `COUNTERFACTUALBINDINGS`.

`ROLEROLEBINDINGS` connect two roles or parameters and assert that, in some descriptive context, they have the same filler. An example would be a `CONTAINERSCHEMA`, which *evokes* a `LOCATIONSCHEMA`, and asserts, via a `ROLEROLEBINDING`, that the container role must be the same as the Location’s relatum role. Simply using classify relations from the two schematic roles to the same entity would lose the information about this constraint, even for other reasoners which might be able to enforce it.

`COUNTERFACTUALBINDINGS` and `CONDITIONALBINDINGS` meanwhile flag that certain role assignments are to be understood as not applying to the real world as some agent might know it, but instead represent situations that may happen or could have happened, and signal that reasoning over these assignments should consider them as such. Conditional bindings are useful therefore to represent branching in task workflows, whereas counterfactual bindings are employed by the counterfactual reasoning we employ to formalize image schemas – e.g., to represent the understanding that a `SUPPORTSCHEMA` also implies that, if the supporter were not there, the supported object would fall.

8.5 Conclusion

We introduce a modeling approach *cum* model that seeks to represent the social interpretation of imperatives, regardless of their linguistic form. The ensuing model is part of the SOMA framework that constitutes a socio-physical model of activities for autonomous robotic agents. As SOMA itself is aligned to the DUL foundational ontology, we showcase where and how our model connects to this axiomatic theory about the high-level domain-independent categories in the real world. Additionally, we detail the employment of our model as an interface between the natural language understanding system and the cognitive infrastructure that drives the robotic agent.

There are obvious limitations to our model, as numerous constraints are outside of the scope of description logics. More expressive models of image schemas and their implications, in first order logic, have been proposed and discussed recently [17, 18]. In other on-going work, we define theories for image schemas in a more tractable formalism of propositional defeasible logic and use them for counterfactual simulation [31]. As future work we are considering a complete reformulation of constructional and schematic knowledge using some form of propositional defeasible logic. Additionally, further reasoning and *mental simulation* is needed to explicate the given instructions completely. This concerns both the grounding of referents as well as the parametrization of the ensuing actions.

As with the work presented herein, we aim to put the insights provided by cognitive linguistic theories to use in making robotic agents more flexible and robust in carrying out the vague and underspecified instructions that are given to them by their human interlocutors. This increase in flexibility is a result of combining physical models that represent actual events, such as a `LOCUTION` with social descriptions thereof, e.g. an `ILLOCUTION`. This two-pronged approach is employed throughout the proposed SOMA-SAY module, whether it is in the `ROLE - OBJECT` or the `TASK - ACTION` pattern. Employing this throughout the cognitive infrastructure of robotic agents enables them to reason about their actions and react felicitously to a given linguistic

input. This work, therefore, represents another step towards moving from robotic agents that can perform a single task to ones that actually master an everyday activity, such as preparing a meal and setting a table.

Chapter 8: References

- [CC1] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98/COLING '98, page 86–90, USA, 1998. Association for Computational Linguistics.
- [CC2] John A. Bateman, Joana Hois, Robert Ross, and Thora Tenbrink. A linguistic ontology of space for natural language processing. *Artif. Intell.*, 174(14):1027–1071, September 2010.
- [CC3] Michael Beetz, Daniel Bessler, Andrei Haidu, Mihai Pomarlan, Asil Kaan Bozcuoglu, and Georg Bartels. Know Rob 2.0 - A 2nd Generation Knowledge Processing Framework for Cognition-Enabled Robotic Agents. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 512–519, 2018.
- [CC4] Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. Cram – a cognitive robot abstract machine for everyday manipulation in human environments. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1012–1017. IEEE, 2010.
- [CC5] Nancy Chang, Jerome Feldman, Robert Porzel, and Keith Sanders. Scaling cognitive linguistics: Formalisms for language understanding. In *Proceedings of the First International Workshop On Scalable Natural Language Understanding*, 2002.
- [CC6] Philipp Cimiano, Paul Buitelaar, Anette Frank, Stefania Racioppa, Michael Sintek, Malte Kiesel, Massimo Romanelli, Berenike Loos, Thierry Declerck, Ralf Engel, Daniel Sonntag, Vanessa Micelli, and Robert Porzel. Linginfo: Design and applications of a model for the integration of linguistic information in ontologies. In *Proc. of OntoLex at LREC*, pages 28–32. ELRA, Mai 2006.
- [CC7] Mihai Pomarlan Sebastian Hoefner John Batemann Rainer Malaka Daniel Bessler, Robert Porzel and Michael Beetz. Foundations of the socio-physical model of activities (soma) for autonomous robotic agents. In *Submitted to EKAW 2020*, 2020.
- [CC8] Scott Farrar and Terry Langendoen. A linguistic ontology for the semantic web. *GLOT International*, 7:97–100, 06 2004.
- [CC9] Jerome Feldman, George Lakoff, David Bailey, Srinu Narayanan, Terry Regier, and Andreas Stolcke. l_0 —the first five years of an automated language acquisition project. *AI Review*, 10:103–129, 1996.
- [CC10] Charles Fillmore. The mechanisms of construction grammar. In *Berkeley Linguistics Society*, volume 14, pages 35–55, 1988.
- [CC11] Charles Fillmore and Paul Kay. *Construction grammar*. CSLI, Stanford, CA, 1999.

- [CC12] Aldo Gangemi, Nicola Guarino, Claudio Masolo, and Alessandro Oltramari. Understanding top-level ontological distinctions. In *International Joint Conference on Artificial Intelligence*, 2001.
- [CC13] Aldo Gangemi and Peter Mika. Understanding the semantic web through descriptions and situations. In *Proceedings of the ODBASE Conference*. Springer, 2003.
- [CC14] Aldo Gangemi, Roberto Navigli, and Paola Velardi. The ontowordnet project: Extension and axiomatization of conceptual relations in wordnet. In Robert Meersman, Zahir Tari, and Douglas C. Schmidt, editors, *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 820–838, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- [CC15] Adele E. Goldberg. *Constructions: A Construction Grammar Approach to Argument Structure*. University of Chicago Press, 1995.
- [CC16] Nicola Guarino and Roberto Poli. Formal ontology in conceptual analysis and knowledge representation. *Special issue of the International Journal of Human and Computer Studies*, 43, 1995.
- [CC17] Maria M. Hedblom, Oliver Kutz, Till Mossakowski, and Fabian Neuhaus. Between contact and support: Introducing a logic for image schemas and directed movement. In *16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017)*, pages 256–268, 2017.
- [CC18] Maria M Hedblom, Oliver Kutz, and Fabian Neuhaus. Image schemas as families of theories. In *Proceedings of the Workshop “Computational Creativity, Concept Invention, and General Intelligence”*, volume 2, pages 19–33, 2015.
- [CC19] Mark Johnson. *The Body in the Mind Metaphors*. University of Chicago Press, 1987.
- [CC20] Jerrold Katz. *Propositional Structure and Illocutionary Force: A Study of PROPOSITIONAL STRUCTURE AND ILLOCUTIONARY Force*. Harvard University Press, Cambridge, Mass., 1980.
- [CC21] George Lakoff. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. University of Chicago Press, 1987.
- [CC22] Ronald W. Langacker. *Foundations of Cognitive Grammar, Vol. 1*. Stanford University Press, 1987.
- [CC23] Berenike Loos and Robert Porzel. Towards ontology-based pragmatic analysis. In *In Proceedings of DIALOR’05*, 2005.
- [CC24] Viviana Mascardi, Valentina Cordì, and Paolo Rosso. A comparison of upper ontologies (technical report disi-tr-06-21). *Dipartimento di Informatica e Scienze dell’Informazione (DISI), Università degli Studi di Genova, Via Dodecaneso*, 35:16146, 2008.
- [CC25] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. Wonderweb deliverable d18, ontology library (final). *ICT project*, 33052:31, 2003.
- [CC26] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. WonderWeb deliverable D18 ontology library (final). Technical report, IST Project 2001-33052 WonderWeb: Ontology Infrastructure for the Semantic Web, 2003.
- [CC27] Till Mossakowski. The distributed ontology, model and specification language–dol. In

International Workshop on Algebraic Development Techniques, pages 5–10. Springer, 2016.

- [CC28] Srini Narayanan. Moving right along: A computational model of metaphoric reasoning about events. In *Proceedings of the Sixteenth National Conference of Artificial Intelligence (AAAI-99)*. AAAI Press, Menlo Park, 1999.
- [CC29] Ian Niles and Adam Pease. Towards a standard upper ontology. In *Proc. of FOIS 2021*, page 2–9, New York, NY, USA, 2001. Association for Computing Machinery.
- [CC30] Johannes Pfau, Robert Porzel, Mihai Pomarlan, Vanja Sophie Cangalovic, Supara Grudpan, Sebastian Höffner, John Bateman, and Rainer Malaka. Give meanings to robots with kitchen clash: A VR human computation serious game for world knowledge accumulation. In Erik D. van der Spek, Stefan Göbel, Ellen Yi-Luen Do, Esteban Clua, and Jannicke Baalsrud Hauge, editors, *Entertainment Computing and Serious Games - First IFIP TC 14 Joint International Conference, ICEC-JCSG 2019, Arequipa, Peru, November 11-15, 2019, Proceedings*, volume 11863 of *Lecture Notes in Computer Science*, pages 85–96. Springer, 2019.
- [CC31] Mihai Pomarlan and John Bateman. Embodied functional relations: a formal account combining abstract logical theory with grounding in simulation. In *Proc. of FOIS 2020*, 2020. under review.
- [CC32] Robert Porzel and John Bryant. Employing the embodied construction grammar formalism for knowledge representation: The case of construal resolution. In *Proc. of 8th International Conference on Cognitive Linguistics*, 2003.
- [CC33] Luc Steels. Basics of fluid construction grammar. *Constructions and Frames*, 9(2):178–225, 2017.
- [CC34] Luc Steels and Joachim De Beule. Unify and merge in fluid construction grammar. In *Proceedings of the Third International Workshop on the Emergence and Evolution of Linguistic Communication (EELC 2006)*, pages 197–223, 2006.

Part IV

Computational Tools

This chapter introduces a number of general tools for building AI systems that understand to some extent descriptions of everyday activities. The tools are applied to the cooking domain.

The first tool is about cutting up a recipe into smaller parts that each represent a single action.
[more]

Chapter 9

Cutting up Recipes

Laura Spillner
Robert Porzel

Digital Media Lab, TZI, University of Bremen
Bremen, Germany

Abstract

As part of a research undertaking that seeks to arrive at the physical execution of household tasks by robots that are given language-based instructions, we take a first step by seeking to cut instructions into activity segments. Textual instructions, e.g. online recipes, are often divided into steps. However, these generally do not map to individual tasks to be carried out by the robotic agent. While pre-trained models and embeddings, which are usually based on declarative texts, have achieved very good performance in other research domains, we find that they struggle with instructive texts. Therefore, we developed a hybrid chunking approach that combines pre-trained models with rule-based understanding, and does not require newly labelled training data. In this work, we compare performance on segmentation of instructive texts into individual tasks between ours and an off-the-shelf baseline approach. We were able to achieve an increase of 11% in the percentage of correctly segmented instructions, 16% in recall and 12% in the F1-score on the task of predicting core actions. In the prediction of the beginning of a new task the increase was 13% in recall and 14% in the F1-score.

Keywords: Natural Language Segmentation, Parsing, Action Extraction

9.1 Introduction

The *rags to riches* narrative is at the heart of many fairy tales and stories and, in a way, is also applicable for the work discussed herein. Most corpora and linguistic analyses contain and examine declarative texts, whereas the text type of instructions has been left relatively neglected to do their work in the scientific basement while the declaratives have a ball.

Nevertheless, digital corpora have emerged in the wild as part of self-help enterprises, for example wikiHow, or as collections of recipes that have been collected and shared at a great variety of websites, e.g. KitchenStories. At the same time the field of household robotics is coming into the focus of researchers as a challenging domain for cognitive agents [1]. As part of the research effort to gain more flexibility these robotic agents will seek to employ textual

instruction - that can be found online - to execute them after performing a series of natural language understanding processes coupled with additional reasoning and inference mechanisms, e.g. mental simulations [20, 15].

Recipes, as other types of instructions, constitute a challenging type of natural language usage. In recipes, in particular, we find established structures, such as the name of the recipe and optionally preparation time, number of servings and tags which are usually followed by an ingredient list. After this the instructions proper are given, using mostly imperative commands, such as:

- (1) Remove from heat and add chopped cilantro.

Several linguistic particularities can be seen already in this example¹: subjects are almost always omitted, and even the direct object can be missing. Furthermore, many words assume unusual parts of speech, as *season* is used as a verb or *cut* can switch between usage as a noun and a verb. Also, classical mass and count noun distinctions become blurred as the countable tomatoes may become liquefied and poured over something else, losing all articles in the process. This makes the employment of out-of-the-box natural language processing solutions that base on pre-trained embeddings, learned from corpora consisting mostly of declarative sentences and maybe some interrogatives, almost impossible, as even state of the art part-of-speech (POS) taggers are thrown off their course by the syntactic idiosyncrasies of instructions.

A correct understanding of such instructions, however, is quite necessary for a robotic agent in order to be able to execute any recipe in a felicitous manner. As would a human reader, such an agent needs to understand the concrete tasks that are embedded in the text, identify entities such as ingredients, utensils and other items mentioned for each task, and combine this with common sense knowledge needed to execute actions which are not explicitly mentioned, e.g. to remove wrappings or open containers before using their contents.

As part of this larger research undertaking - to go from language-based instructions to their physical execution by robots - we take a first step by seeking to cut instructions into activity-specific segments. Textual recipes, as found online, are often divided into steps, however, these generally do not map to individual actions which would correspond to specific tasks to be carried out by the robotic agent. Often, a step or even a sentence groups together several actions. This grouping is certainly adequate for a human reader, but actually requires multiple distinct activities, such as first chopping something and then adding it to the pan. Therefore, the first challenge in understanding instructive texts is to segment them into individual tasks.

In the following section, we will therefore look at prior work on understanding instructions via natural language processing methods. Thereafter, we will showcase our own approaches to segment recipes into sizable chunks in order to create structured corpora. These can be employed as benchmarks for this particular challenge as well as input into subsequent processing modules that further explicate and seek to *understand* the individual segments in a given context.

9.2 Related Work

Segmenting natural language instructions into an ordered set of individual tasks might seem to be a natural first step in understanding any how-to text. Yet, it appears to be a step that is often skipped: Prior work on recipe understanding aims to parse on a sentence level in order to predict which of the individual actions relate to which of the mentioned tools or

¹All examples are taken from KitchenStories.com.

ingredients [6, 17]. Others employ whole sentences as tasks or actions, and ignore cases where a sentence includes more than one action [10]. Our notion of a *task* comes with a specific level of granularity². We consider a task to constitute an action chunk that can be empirically distinguished from the next, e.g. using hotspot detection based on biosignals and think-aloud data from humans [11, 12].

Generally speaking, segmenting instructions into task-specific chunks is not dissimilar to the problem of sentence segmentation. This is usually achieved with grammatical rules. There are some cases, however, where these approaches do not perform well, e.g. in the case of some particular languages as well as depending on the level of formality employed [19]. Using supervised training methods to predict the beginning and end of tasks in instructive texts would require expensive labelled training data; as does fine-tuning pre-trained word embeddings such as BERT - a method that has achieved great results in many other natural language processing problems - for imperatives.

In some cases, it is possible to circumvent the problem of segmenting into specific tasks entirely. For example, the problem of recipe understanding can be approached as a clustering task that aims to group recipe instructions according to the type of event or action found in the instruction, corresponding to the kind of commands a robotic agent would execute in a kitchen domain [21]. In this approach, each sentence is viewed as one task. While the authors recognized that one sentence can mention several actions, they solved this problem by allowing a sentence to be included in more than one cluster.

Other approaches treat the problem even more pragmatically. Aiming to understand affordances of objects in the kitchen domain, Losing *et al.* classify tokens as actions, tools, or objects [10]. To do this, they view each sentence as one task, and only take those sentences into account where one of each of those three classes of tokens can be found. If for example more than one action is found in a sentence, the first one is chosen - while they note that this might lead to incorrect sets of actions and tools, they argue that this limitation is outweighed by the advantages gained from a more cost-efficient method.

Several other works use a similar approach. For example, Pustejovsky *et al.* view a *cooking event* as an action which is related to a number of entities, such as ingredients or props. While one sentence can mention several actions, utensils or ingredients, this notion of a cooking event corresponds very well to what we refer to as a *task*. Pustejovsky *et al.* use this model to annotate a larger corpus of recipes, based on which they tackle downstream tasks such as event ordering and aligning text-based recipes to videos. They annotated tokens manually as actions, ingredients, tools, habitats or containers, and the relations between them [17]. These kinds of tasks can also be constructed using entity recognition and relation prediction [6]. Jiang *et al.* use this method to create a corpus of annotated recipes. They view recipe understanding as an entity recognition task, differentiating between actions, tools, and ingredients, among others. While their model shows very good results in entity recognition with a micro F1-score of around 90%, correctly predicting relations between these entities appears to be more difficult, with a micro F1-score of about 67%.

While this method is certainly promising, the results from the prior work discussed herein show that correctly relating actions to their necessary tools is still a very difficult task [6]. Evaluating the annotation on the task of event ordering yielded very good results in ordering two sentences, but accuracy drops to around 62% when trying to order three or more sentences [17]. Since we are interested in eventually using this corpus to understand affordances and tools used for

²At a coarser level *setting the table* or *making supper* can be considered tasks, and - for a robotic agent - even finer levels, e.g. grasping a cup or moving it from A to B, can be viewed as challenging tasks themselves.

specific tasks in the kitchen domain, we instead want to divide a recipe into tasks based on the structure of the text, aiming to use this corpus to train models for downstream tasks such as tool prediction.

Nevertheless, segmenting a recipe into tasks can be viewed as a somewhat similar problem as the segmentation of texts into sentences. In many cases, this problem can be quite trivial, e.g. for standard European languages text can be segmented into sentences with a list of sentence-ending punctuation marks and a list of exceptions, such as abbreviations, where a period is used but does not signify the end of a sentence. Palmer *et al.* noticed that there are a number of dependencies that complicate the task: multi-lingual texts might use many different punctuation rules, and other types of texts might stray from typical conventions of the language [14]. Informal language often includes non-standard punctuation, capitalization or spelling, omits sentence-ending punctuation in individual messages or at the end of paragraphs, and uses emoticons or multiple punctuation marks. In this case, statistical methods are generally more appropriate [19].

In a recent study concerning post card texts, a CRF model is trained to classify tokens according to the IOB format, thereby treating sentence segmentation as a chunking task [19]. In this work the authors were able to show that their model achieved significantly better results than classic unsupervised methods based on abbreviations [8].

While this method could be applied to recipe instructions as well, a drawback is the training data required for supervised model training. We examined another problem domain that is similar to that of recipe instructions: Customer reviews often mention several aspects of the product or service in one sentence, and in order to understand how a customer rates a product in these categories, it is necessary to segment sentences into individual chunks for each aspect mentioned.

Zhu *et al.* proposed a multi-aspect segmentation model which segments sentences from customer reviews based on which aspects of a product or service are mentioned [22]. This model was later improved by a different team [13]. Their IMAS algorithm scores customer opinions in different aspects by first segmenting sentences into chunks depending on punctuation and conjunctions, and then predicting an aspect for each chunk.

Based on these approaches and findings, we explored a hybrid chunking approach in this work. We utilized pre-trained word embeddings to achieve a general understanding of the grammatical structure of a sentence, and used grammatical rules to split it into individual task-based segments. In the following we describe how we created a corpus of segmented recipes using this method. The evaluation of our method and possible error categories provides some interesting insights on how pre-trained solutions fail when dealing with imperative sentences, and how this can be remedied with rule-based enhancements.

9.3 The Hybrid Chunking Approach

Ultimately, the goal in recipe understanding for robotic agents is to convert recipes from more or less formal and more or less structured natural language into a set of tasks. Each task should include all the information necessary to execute it; that is, at the very least an action that the agent has to perform, e.g. cutting, the objects that are needed to do this, e.g. an appropriate knife and a supporting surface such as a cutting board, as well as any additional information on the quality of the action, e.g. the thickness of the slice.

In many cases, some of this information will be implicit or missing and assumed common sense knowledge, such as the typical thickness of bread slices, while other information is found

somewhere in the surrounding text, for example, an ingredient or tool mentioned previously. Some common sense actions might also not be explicitly mentioned in the recipe, e.g. turning on the stove or taking the bread out of the box. Inferring these actions requires detailed knowledge models as well as planning of actions by robotic agents.

To make matters worse, some tasks are merely implied, but not phrased as such, and can be seen as hidden tasks, e.g. “to add melted butter” implies to perform a “melting the butter” task beforehand. For now, we disregard these kinds of hidden tasks and focus only on those that are explicitly mentioned in the recipe. We view a task as a phrase prompting the reader to do something, meaning that a task must at the very least consist of a verb, and be in the imperative form.

Many online recipes are already structured into steps, often defined in formats such as RecipeML³. However, how this structure is applied differs greatly depending on the author of the recipe: some use one sentence per step, others whole paragraphs where a recipe is only made up of three or four steps total. Generally, one step in RecipeML does not map to one instructional task. Therefore, we can disregard these internal structures of the recipe and instead view all recipes as a collection of sentences without further subdivisions.

The first step is then to segment the body text of the recipe into sentences, and further segment these sentences into individual tasks. For this, we assume that an individual task must include at least a verb and cannot span several sentences. Also, each sub-step that can be described by a verb constitutes its own task. As mentioned above, there is always some additional information which is necessary for the execution of an action, but that is not included in the task. Often this is implicit common sense information, such as tools that are needed to complete a task, as in Example (2) both a suitable knife and a cutting board are implicit:

(2) Cut the tomatoes.

In other cases, necessary information such as ingredients are found in the surrounding text, and an agent needs to be capable of resolving zero anaphora in order to fully understand an instruction.

(3) Roughly chop scallions and add to the bowl.

For us, Example (3) constitutes two tasks: a) to “Roughly chop scallions” and b) to “Add [them] to the bowl”. As the pronoun is not given, we consider it to constitute a zero anaphora pointing to the previously chopped scallions. Another, similar problem is created by splitting one sentence which coordinates two actions in a row into two separate tasks:

(4) Peel and grate the carrot.

Example (4) includes the tasks “Peel [the carrot]” and “Grate the carrot”, but splitting the sentence in two creates an ellipsis: “Peel”. In cases like this, the split adds a kind of zero anaphora that was not needed in the original text, which used a coordinative construction to connect the two conjoins. This makes understanding the task more difficult for the agent. However, splitting instructions that include multiple tasks also makes it easier to understand which ingredients or tools belong to which action. We believe this will be helpful in understanding cooking tasks, as predicting relations between cooking entities is an area where other approaches are still struggling [6].

In a way, this shifts part of the problem. Other methods predict actions and entities on a sentence level to predict relations between those and thus require zero anaphora resolution for sentences as given in Example (3) [6, 17]. Instead, our approach is to segment the sentences

³See <http://www.formatdata.com/recipeml>

into tasks first. This has the advantage of simplifying relation prediction in many cases, since for those tasks we can safely assume that all entities mentioned in a task relate to the action it is describing. It, however, causes the disadvantage of creating ellipses that are more difficult to resolve. Nevertheless, this would not be a problem for approaches where the resolution of zero anaphora or explication of implicit information is not needed.

9.3.1 Dataset

The dataset used for this study consists of recipes hosted on the KitchenStories website, which are uploaded by users and provided online free of charge. The recipes all follow a similar format. They usually consist of 3-5 steps, whereby each step is comprised of several sentences. They provide an overall list of ingredients as well as list ingredients and tools used in each step. As the content is user generated, the language used in the recipes is quite diverse and sometimes contains misspellings or non-standard language. Additionally, the tool and ingredient lists are not always complete.

The original corpus - which was constructed for an unrelated tool prediction approach - is made up of the first 1000 recipes that were available in the recipe section of the website at that point. During pre-processing, recipes were structured as lists of sentences using a rule-based sentence segmentation approach followed by some manual corrections. Some sentences and parts of sentences were removed, namely those that provide additional information not necessary to the recipe itself and do not represent instructions as they usually are references to specific video guides. Additionally, the sentences were adjusted to all end with a period, which were missing in some cases. We also removed special characters such as emojis.

From this dataset, 600 sentences were used for this work on task chunking. We split the set into 100 sentences which were used as a *training set*, on which we initially tested the baseline approach and based on which we formulated and refined the rules used in the hybrid chunking approach. We then evaluated both approaches on the *test set* of 500 sentences. The sentences were manually annotated by splitting them into tasks where necessary and labelling a so-called *action verb* that is the main verb of the imperative sentence, of which there can be only one per task. Based on that, the test set was restructured to have two labels per token. One label denotes whether or not the token is an action verb. The other label denotes if the token is the beginning of (B), inside (I), or outside (O) of a task.

9.3.2 Baseline

As our goal was to find a method for segmentation that is not too computationally expensive and does not require extensive amounts of manually-labelled training data, we need some baseline for evaluation against such a comparable “out-of-the-box” approach. For this baseline approach, we employed the spaCy toolkit [5], which is an open source natural language processing (NLP) library that provides a number of different pre-trained word embeddings, as well as a completely pre-trained processing pipeline that features a tokenizer, POS tagger, a parser that constructs dependency trees, as well as named entity recognition and other common downstream tasks. Not only is spaCy a common tool in the industry, it has also been used by many research projects, especially for pre-processing tasks such as tokenizing, sentence segmentation and POS tagging, on the basis of which more sophisticated models are being trained [7, 9, 18]. Therefore, we want to investigate how well spaCy - as an out-of-the-box tool - performs for these kinds of tasks in the context of instructions and imperative sentences, on which it has not been trained.

SpaCy itself does of course not provide a pre-trained method for instruction segmentation. However, the problem can easily be reframed to be about the grammatical structure of a

sentence: As each task is to consist of exactly one *action verb*, this means that a task will consist of an independent clause as well as whatever dependent clauses relate to it. If a sentence consists of several independent clauses, it should be split into individual tasks based on those.

SpaCy provides a pre-trained dependency parser which constructs a dependency tree of a given sentence. Generally, the predicate is at the root of a sentence, and each clause has its own subtree. If a sentence consists of more than one independent clause, they will be connected by conjunctions, and each subtree will have the predicate of its main clause at the root. Thus, in theory, a task corresponds to a subtree that is headed by a verb tagged as *VB* (base form) or *VBP* (non-3rd person singular present), which is either the root of the sentence or a direct child of the root of the sentence, in cases when there is more than one main clause. Of course, this assumes that there are no mistakes in the POS tagging or dependency parsing. Based on this, we constructed a baseline model that classifies tokens as action verbs depending on their POS and dependency tag, and splits sentences into tasks based on the dependency tree of the sentence, where each task corresponds to the subtree headed by its action verb.

In practice, when there is an error in the predicted POS tag of a token, this often leads to mistakes in the dependency tree. Particularly when it comes to instructions in the imperative form, it leads to problems in the predicted verb form as well as the sentence structure; likely because these components of spaCy’s pipeline were mainly trained on declarative sentences.

We also noticed that the domain of recipes poses an additional challenge in POS tagging: there is a large number of words that can be used either as a verb or as a noun depending on the context, e.g. salt, pepper, juice, zest, dollop, spread, etc. - even *garlic* can be used as a verb. Combined with the structure of non-declarative sentences, this leads to a significant number of errors when it comes to nouns and verbs. Based on these observations we made when analyzing the first 100 sentences of our dataset that underwent this baseline approach, we developed a new hybrid approach combining pre-trained embeddings and a small set of additional rules.

9.3.3 Rule-based Understanding

Given that using spaCy’s pre-trained model out-of-the-box will not work very well for the domain of recipes, there are a number of ways to approach the problem. All of the components of spaCy’s pipeline can be retrained on a domain-specific dataset. Additionally, we can go one step further and also train the word embeddings that serve as input features for the NLP pipeline on our recipe corpus. However, this requires a lot of labelled recipe data for supervised training, and training new embeddings is computationally expensive.

Instead, we considered the information that is already included in the out-of-the-box models in combination with our knowledge of the structure of imperative sentences. We employed three main strategies to improve upon spaCy’s baseline performance: firstly, we improved POS tagging by automatically converting the instructions in imperative form into declarative sentences that should be easier for the model to parse. Secondly, instead of using the final predictions of the pre-trained POS tagger, we called the prediction method directly in order to weigh all possible predictions. This makes it possible to look at the probabilities of all individual tags. Thirdly, we disregarded the dependency tree, and instead split the sentences into tasks by first dividing it into chunks and then recombining those into tasks based on the likelihood of the combined chunks constituting a complete sentence.

Similar to established methods employed for aspect segmentation [22, 13], we split each sentence at each possible delimiter, such as commas, semicolons, *and*, or *or*. Some of the resulting chunks constitute individual independent or dependent clauses, while others are simply part of enumerations. The main idea behind our approach is to find those *core chunks* that stand as

main clauses by themselves as they include an action verb. All other so-called *satellite chunks* are appended to the core chunk that they most likely belong to, e.g.:

- in the case of enumerations that means the main clause they were part of in the first place.
- in the case of dependent clauses that means the main clause they relate to.

If there is only one main clause and one action verb in a sentence, the case is trivial - the entire sentence constitutes one task. Otherwise, appending satellite chunks to their respective core chunk requires predicting which combinations are the most likely ones.

When trying to identify which of the chunks of a sentence are core chunks, we also employ the other two strategies mentioned above. As imperative sentences are missing the subject of the sentence, the implied “you” is added as a word to the beginning of a chunk before parsing it with spaCy. Since the sentence was split into chunks first, the subject would necessarily be at the beginning of a chunk if it includes a predicate. Then, all respective probabilities of all possible tags for the other tokens are analyzed. Generally, the verb would be expected to come in second place after the subject, however, there might be any number of adverbs between the two (which were at the start of the original chunk); therefore later tokens also have to be considered - if there are several possible action verbs, the first one is chosen. Furthermore, we take into account second- and third-choice POS tags, and include other possible verb tags in addition to *VBP* (non-third person singular). We consider tokens as possible action verbs if:

- the most likely tag is *VBD* (past tense) or *VBZ* (third person singular present), if the second-choice tag is *VBP*
- nouns if the second-choice tag is *VBP* and the probability of the noun tag is less than 65%
- the sum of the probabilities of all possible verb tags exceeds 85%

The thresholds for these conditions were chosen experimentally.

All the chunks where no action verb was found must then be satellite chunks, either relative clauses without a predicate or a part of an enumeration. We combine these with one of the core chunks, by recursively testing whether joining a satellite chunk to its predecessor or successor improves the overall confidence in the POS tagging probabilities. This will result in the sentence being segmented into tasks, where each task includes exactly one action verb.

9.3.4 Empirical Examination

To analyze the performance of this hybrid chunking approach in comparison with the out-of-the-box baseline method, we evaluated both approaches on the previously manually labelled dataset of 500 sentences. This was done by looking both at the classification of action verbs as a binary classification task as well as the segmentation into tasks. The reason for this is two-fold:

Firstly, it is necessary in order to evaluate our method in comparison to related works that employ supervised trained models to classify action verbs but do not split recipes into tasks. Secondly, it allows us to differentiate between segmentation errors that are based on incorrectly recognized action verbs and those where the segmentation algorithm failed in reconstructing sentences from their chunks.

In order to evaluate the segmentation, we calculated both a sentence based accuracy, i.e. checking for each sentence whether or not it was split correctly, as well as a token-level score using

the IOB tags. Thus, it could later be possible to do a detailed comparison of our results with models trained on labelled training data in the same format in a manner that is analogous to prior work on sentence segmentation [19].

9.4 Results

Approach	Accuracy	Precision	Recall	F1-score
Baseline	0.96	0.88	0.79	0.83
Hybrid Chunking	0.99	0.95	0.95	0.95

Table 9.1: Comparison of action verb prediction results

We evaluated the baseline model using spaCy’s dependency parser as well as our hybrid model on 500 manually labelled sentences that had been collected from KitchenStories recipes. First, we detail how well both approaches perform in classifying action verbs as well as segmenting the sentences into tasks. Then, we discuss where the original, spaCy-based out-of-the-box model breaks down, which errors can be resolved with our approach and which errors persist.

9.4.1 Classifying Action Verbs

Classifying tokens as action verbs is a binary classification task. Out of 500 sentences, 250 contain only one action verb. On average, there are 1.652 action verbs per sentence; with 826 out of a total 7574 tokens being action verbs. Using spaCy’s POS tagger, the baseline approach achieves an accuracy of 0.96 and an F1-score of 0.83. In comparison, the hybrid approach performs better, achieving an accuracy of 0.99 and an F1-score of 0.95. As shown in Table 9.1, the difference is mainly due to worse recall in the baseline - 21% of the action verbs are missed by the baseline approach.

In theory, recognizing action verbs with spaCy should be trivial: given our definition of action verbs, action verbs should be the predicates of their clause, i.e. the root of a clause in the dependency tree. SpaCy’s pre-trained models have achieved very good performance in other domains, reporting an accuracy of 97.4% for dependency parsing [5]. We argue that the model performs worse in this case due to the domain-specific imperative sentence structure.

Using the hybrid approach, the ratio of false positives and false negatives is equal (0.5), and both precision and recall equal 0.95. In comparison, the baseline approach leads to a larger percentage of false negatives, which in this case are action verbs that were missed. These constitute 65% of all action verb mistakes.

Jiang *et al.* also aimed to predict actions in recipes, among other entities such as ingredients and tools. They trained a feed forward neural network on a labelled recipe corpus, and achieved an F1-score of 0.97 for action tokens. Thus, our hybrid chunking approach was able to achieve performance that is almost on par with supervised trained models. We were able to show that by including grammar-based rules and taking domain knowledge into account, we can improve performance while still using pre-trained embeddings and models, without requiring new training data specific to the domain.

9.4.2 Segmenting Tasks

We evaluated task segmentation both on a per-sentence basis (counting only whether a sentence was segmented correctly or not), as well as on a per-token basis as a multiclass classification task, where tokens had to be classified according to the IOB scheme.

	Segmentation	Token (IOB)	Prediction of Beginning of Task		
Approach	Accuracy	micro F1-score	Precision	Recall	F1-score
Baseline	0.74	0.92	0.92	0.83	0.88
Hybrid Chunking	0.85	0.97	0.91	0.97	0.94

Table 9.2: Results of Sentence Segmentation

The baseline model was able to segment 371 out of 500 sentences in the corpus correctly, achieving an accuracy of 0.74. Out of the incorrectly segmented sentences, 88% were sentences which had more than one action verb. In comparison, the hybrid approach improved upon the baseline performance with an accuracy of 0.85 (segmenting 424 sentences correctly). It especially improved when it comes to sentences with more than one action verb: the hybrid approach was able to segment 81% of those sentences correctly, compared to only 54% with the baseline approach. 63% out of the sentences that the hybrid approach classified incorrectly were sentences with more than one action verb.

The results of the token-based evaluation also show a discrepancy between the two approaches. Firstly, we calculated micro F1-scores⁴ for the baseline and the hybrid approach: The out-of-the-box baseline approach achieved a micro F1-score of 0.92, while the hybrid approach performed better with a micro F1-score of 0.97.

Secondly, we compared specifically how well both approaches performed in recognizing the beginning of individual tasks, thereby ignoring correct labelling of tokens in between two tasks, which are usually delimiters such as commas or *and*. While there is no noteworthy difference in the precision of 0.92 for the baseline and 0.91 for the hybrid approach, the hybrid approach again performs better in terms of recall (0.97) *versus* the baseline (0.83), and thus achieves an overall better F1-score (0.94 *versus* 0.88). This shows that the baseline approach yields more false negatives, where the system does not recognize the beginning of a new task. We looked at the confusion matrix of how tokens of each label class were classified, and saw that 15% of tokens at the beginning of a task are incorrectly classified as being inside a task, while the same happens for only 3% with the hybrid model.

9.4.3 Discussion of Error Types

Most cases of erroneous segmentations of sentences into tasks were a result of incorrectly identified action verbs. This was the case for 84% of all errors for the baseline approach, while only 62% of errors for the hybrid approach were the result of action verb mistakes.

Theoretically, an action verb should be the predicate of a sentence, in simple present and in non-3rd person singular form. However, sometimes the pre-trained POS tagging model misidentifies the form of a verb, or recognizes a verb as a noun or *vice versa*. In the baseline approach, most of the mistakes in identifying action verbs are false negatives, that is, an action verb is not recognized as such, which constitute 65% of cases. The remaining 35% are mistakes where another word - either a verb that is not the predicate, or in some cases a noun - is incorrectly identified as the action verb.

Out of those false negatives, most are cases where the baseline model is not able to correctly

⁴For this task, each token has to be classified in one of the three classes. Tokens that are false positives for one class are also false negatives for another class. *Ergo* the sum of all false negatives equals the sum of all false positives, equals the number of all incorrectly classified tokens. Thus, the micro-averaged precision and micro-averaged recall both equal the accuracy over all classes, i.e. percentage of correctly classified tokens, which then also equals the micro F1-score.

identify the form of the verb or identify the verb as the predicate of the clause - likely due to the idiosyncratic structure of the imperative sentences. In order to remedy this in the hybrid approach we discarded spaCy’s prediction of the predicate in the dependency tree, and instead looked at several of the possible verb tags and their respective probabilities instead of only the most likely one. With this strategy, we were able to reduce the number of false negatives in the action verbs from 21% to 5%.

Additionally, there are some cases where an action verb is incorrectly classified as a noun. This usually happens with verbs that are more commonly used as nouns, e.g. *juice*. Because of this, we also considered tokens which were tagged as nouns, if the probability was rather low. However, this clashes with the opposite problem where nouns are incorrectly recognized as verbs.

Nouns being incorrectly recognized as verbs is the most common kind of false positive error we observed, which accounted for 35% of all action verb mistakes with the baseline approach. While many of the errors are very reasonable, as *salt* or *pepper* could easily constitute a verb in a recipe text, others were more surprising, e.g. that *garlic* is also used as a verb in English. While a list of exceptions could be useful here, many words are those that are used often as a verb or a noun, such as *salt*. Furthermore, discarding possible nouns has the risk of increasing the number of false negatives for the same kind of words, if too many options are discarded. There are some other cases where other forms of verbs are incorrectly identified as action verbs; usually verbs in past tense. Many of these cases can be caught by checking for a suffix of *-ed* and testing whether the word remains a recognizable verb when it is removed. With these strategies, we were able to reduce the share of false positives out of all predicted action verbs from 12% with the baseline to 5% with the hybrid approach.

Overall, there was a greater reduction in false negatives than in false positives, which is explained by the need to be careful not to discard too many actual verbs as possible nouns. Thus, the ratio of false negatives and false positives with the hybrid approach is 0.5 for each.

While mistakes in sentence segmentation are generally caused by false positive or false negative action verbs, there are some cases where sentences were segmented incorrectly even when all action verbs in the sentence were classified correctly.

For the baseline approach, 84% of the incorrectly segmented sentences were sentences where the action verb was not identified correctly. Only 21 sentences were segmented incorrectly despite a correctly recognized action verb. With the hybrid approach, there were 29 sentences which were segmented incorrectly in spite of correctly recognized action verbs, making up 38% of all incorrectly segmented sentences. This implies that the hybrid approach mainly improves the recognition of action verbs, while mistakes caused by incorrectly connected chunks persist and even slightly increase compared to the baseline.

In analyzing these incorrectly segmented sentences, we noticed that the reasons that this happens differ significantly between the two approaches. For the baseline approach, the incorrect segmentation usually happens when the entire dependency tree of the sentence is incorrect; that is, the dependency analysis is completely wrong in these cases, leading to tasks that are nonsensical. With the hybrid approach, the most common occurrence is that the sentence was segmented into chunks correctly, but a satellite chunk is added to the wrong core chunk, which means that necessary information for one task is found in another one where it does not make sense. While the mistakes using the hybrid approach appear less severe, they in turn happen somewhat more often than in the baseline approach.

Finding remedies for these kinds of mistakes is more difficult than for action word classification. While the incorrect dependency analysis is mainly a grammatical problem, combining the

correct chunks with each other sometimes requires not solely grammatical but common sense knowledge as well, which could potentially be inferred using additional ontological models of the domain [3].

9.5 Conclusion

Textual instructions such as recipes, which exist in large user-generated corpora online, can be used to aid robotic agents in understanding and executing household tasks. This paper presents a first processing step towards this goal, focusing on segmenting natural language instructions into individual, robot-executable tasks. While pre-trained models from NLP tools such as spaCy perform well in analyzing natural language texts in other domains, we have shown that they struggle when it comes to instructions. We implemented a baseline approach using spaCy’s POS tagger and dependency parser and evaluated this approach on the tasks of predicting action verbs in instructive sentences and segmenting sentences into tasks. However, this implementation segmented 26% of sentences incorrectly. The problem mainly stemmed from an imperfect recall of action verbs as well as false determinations of the beginning tokens of new tasks: the baseline approach missed 21% of action verbs and 17% of beginning tokens.

Based on a number of error types identified from the baseline approach, we implemented a hybrid chunking approach which combines the same pre-trained word embeddings with rule-based understanding. We evaluated this in comparison to the baseline, and found that it is indeed possible to improve performance without requiring new training data or domain-specific models. This was achieved by taking into account the grammatical structure of imperative sentences as well as the relative probabilities of POS tags that were discarded by the pre-trained model. Our approach was able to achieve noticeably higher recall on both tasks, missing only 5% of action verbs and 3% of tokens beginning a new task. It therefore segmented 85% of all sentences correctly into individual tasks.

An analysis of the remaining types of mistakes shows that additional semantic resources could become useful for attaching satellite chunks to their rightful cores. Especially formalized common sense knowledge, for example concerning the affordances of used objects [3] and the internal structure of actions [2] can be used to apply ontology-based scoring algorithms that operate on said graphs [4]. This could also be enhanced by using semantic parsers that already produce ontology-based semantic specifications from texts [16], but operate on a sentence level and not on the level of an individual task.

The work presented herein, of course, is only a minimal but necessary first step towards making recipes understandable for robots. The ensuing ensemble of challenges ranges from the zero anaphora resolution to the explication of additional implicit information for which approaches using human computation constitute a feasible direction [15]. Both the problem of explicating implicit information as well as the need for physical grounding could, in turn, benefit from a previous task chunking step, based on a divide and conquer strategy. In modern cognitive architectures for robotic agents [1], action selection takes the center stage and becomes more flexible in the overall endeavor to create cognitive agents that are not programmed to perform individual actions in constant environments, but have a general competence to sequence individual tasks in an appropriate manner.

Therefore, we will use open platforms, such as Kaggle, to publish annotated corpora in order to create benchmarks that can be used by the research community to compare different approach with each other. We, thereby, hope to contribute to get the linguistically challenging domain of instructions out of its current basement position into the light of scientific scrutiny.

Chapter 9: References

- [DA1] Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. CRAM – A Cognitive Robot Abstract Machine for everyday manipulation in human environments. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1012–1017, Taipei, 2010. IEEE.
- [DA2] Daniel Beßler, Robert Porzel, Mihai Pomarlan, and Michael Beetz. Foundational models for manipulation activity parsing. In Timothy Jung, M. Claudia tom Dieck, and Philipp A. Rauschnabel, editors, *Augmented Reality and Virtual Reality*. Springer, 2019. 978-3-030-37869-1.
- [DA3] Daniel Beßler, Robert Porzel, Mihai Pomarlan, Abhijit Vyas, Sebastian Höffner, Michael Beetz, Rainer Malaka, and John Bateman. Foundations of the Socio-physical Model of Activities (SOMA) for Autonomous Robotic Agents. In *12th International Conference on Formal Ontology in Information Systems*, 2021.
- [DA4] Iryna Gurevych, Rainer Malaka, Robert Porzel, and Hans-Peter Zorn. Semantic coherence scoring using an ontology. In *Proceedings of the HLT-NAACL Conference*, Edmonton, Canada, 2003.
- [DA5] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. *spacy: Industrial-strength natural language processing in python*, 2020.
- [DA6] Yiwei Jiang, Klim Zaporozhets, Johannes Deleu, Thomas Demeester, and Chris Develder. Recipe instruction semantics corpus (RISeC): Resolving semantic structure and zero anaphora in recipes. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 821–826, Suzhou, China, December 2020. Association for Computational Linguistics.
- [DA7] Yuval Kirstain, Patrick Lewis, Sebastian Riedel, and Omer Levy. A few more examples may be worth billions of parameters. *arXiv preprint arXiv:2110.04374*, 2021.
- [DA8] Tibor Kiss and Jan Strunk. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525, 2006.
- [DA9] Stephen Wai Hang Kwok, Sai Kumar Vadde, and Guanjin Wang. Tweet topics and sentiments relating to covid-19 vaccination among australian twitter users: Machine learning analysis. *J Med Internet Res*, 23(5):e26953, May 2021.
- [DA10] Viktor Losing, Lydia Fischer, and Jörg Deigmöller. Extraction of common-sense relations from procedural task instructions using BERT. In *Proceedings of the 11th Global Wordnet Conference*, pages 81–90, University of South Africa (UNISA), January 2021. Global Wordnet Association.
- [DA11] Moritz Meier, Celeste Mason, Robert Porzel, Felix Putze, and Tanja Schultz. Syn-

- chronized multimodal recording of a table setting dataset. In *Proceedings of the IROS 2018 Workshop on Latest Advances in Big Activity Data Sources for Robotics & New Challenges (Madrid)*, 2018.
- [DA12] Moritz Meier, Celeste Mason, Felix Putze, and Tanja Schultz. Comparative analysis of think-aloud methods for everyday activities in the context of cognitive robotics. In *20th Annual Conference of the International Speech Communication Association*, Graz, Austria, October 2019. Interspeech 2019.
- [DA13] David Millie, William A. Young II, Naime F. Kayaalp, Gary R. Weckman, and Can Celikbilek. Extracting customer opinions associated with an aspect by using a heuristic based sentence segmentation approach. *International Journal of Business Information Systems*, 26:236, 2017.
- [DA14] David D Palmer. Tokenisation and sentence segmentation. *Handbook of natural language processing*, pages 11–35, 2000.
- [DA15] Johannes Pfau, Robert Porzel, Mihai Pomarlan, Vanja Sophie Cangalovic, Supara Grudpan, Sebastian Höffner, John Bateman, and Rainer Malaka. Give MEANinGS to Robots with Kitchen Clash: A VR Human Computation Serious Game for World Knowledge Accumulation. In Erik van der Spek, Stefan Göbel, Ellen Yi-Luen Do, Esteban Clua, and Jannicke Baalsrud Hauge, editors, *Entertainment Computing and Serious Games*, volume 11863, pages 85–96. Springer International Publishing, Cham, 2019.
- [DA16] Robert Porzel and Vanja Sophie Cangalovic. What Say You: An Ontological Representation of Imperative Meaning for Human-Robot Interaction. In *Proceedings of the JOWO – Ontology Workshops*, Bolzano, Italy, 2020.
- [DA17] James Pustejovsky, E. Holderness, Jingxuan Tu, Parker Glenn, Kyeongmin Rim, Kelley Lynch, and Richard Brutti. Designing multimodal datasets for nlp challenges. *ArXiv*, abs/2105.05999, 2021.
- [DA18] Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. *arXiv preprint arXiv:2104.06644*, 2021.
- [DA19] Kyoko Sugisaki. Word and sentence segmentation in german: Overcoming idiosyncrasies in the use of punctuation in private communication. In Georg Rehm and Thierry Declerck, editors, *Language Technologies for the Challenges of the Digital Age*, pages 62–71, Cham, 2018. Springer International Publishing.
- [DA20] Moritz Tenorth and Michael Beetz. Representations for robot knowledge in the KnowRob framework. *Artificial Intelligence*, 247:151–169, 2017.
- [DA21] Filippas Ventirozos, Mauricio Jacobo-Romero, Sarah Clinch, and Riza Batista-Navarro. Interactive clustering of cooking recipe instructions: Towards the automatic detection of events involving kitchen devices. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, pages 341–346, 2021.
- [DA22] Jingbo Zhu, Huizhen Wang, Muhua Zhu, Benjamin K. Tsou, and Matthew Ma. Aspect-based opinion polling from customer reviews. *IEEE Transactions on Affective Computing*, 2(1), 2011.

Chapter 10

Integrative Narrative Networks

Lara Verheyen*, Luc Steels[†]

*Artificial Intelligence Laboratory, Vrije Universiteit Brussel

[†]Venice International University

Abstract

When trying to cope with complex situations and tasks, we can often rely on ready-made solutions based on past experiences. For example, anyone who has cooked spaghetti a couple of times no longer needs instructions or a cookbook and immediately knows what to do the next time they want to make their favorite pasta dish again. The acquisition and application of these ready-made solutions is currently approached in AI using large neural and statistical models acquired from massive amounts of data and data-driven methods such as deep learning. These methods are not (directly) concerned with meaning but perform feature-detection to retrieve past solution patterns and then apply these patterns to handle similar cases. The power but also the limitation of these statistical methods is well illustrated with ChatGPT and other Large Language Models [5].

Keywords: Narrative Networks, Language Understanding, Metrics

10.1 Introduction

In text-processing this pattern recognition approach is known as **distant reading** [17]. Distant reading refers to the use of methods from information retrieval, data mining and statistics to find information or regularities in perceptual observations or texts without actually being concerned with meaning. Distributional semantics, word embeddings and the counting of token frequencies are all examples of methods developed to operationalize distant reading. The term distant reading extends to situation awareness, image interpretation, action selection or other tasks that can be solved by pattern-recognition and pattern-application.

In contrast, there are many cases where there is not enough prior experience and the agent has to go beyond statistical models to develop deeper content models. For example, the diagnosis of a disease can be done using distant reading if the disease is very common and very well known but otherwise it requires a deeper causal model of biological processes grounded in molecular biology and sophisticated instrumentation. In the case of text-processing, this process is called **close reading** [17] and we can apply this term again more generally to other inputs beyond text.

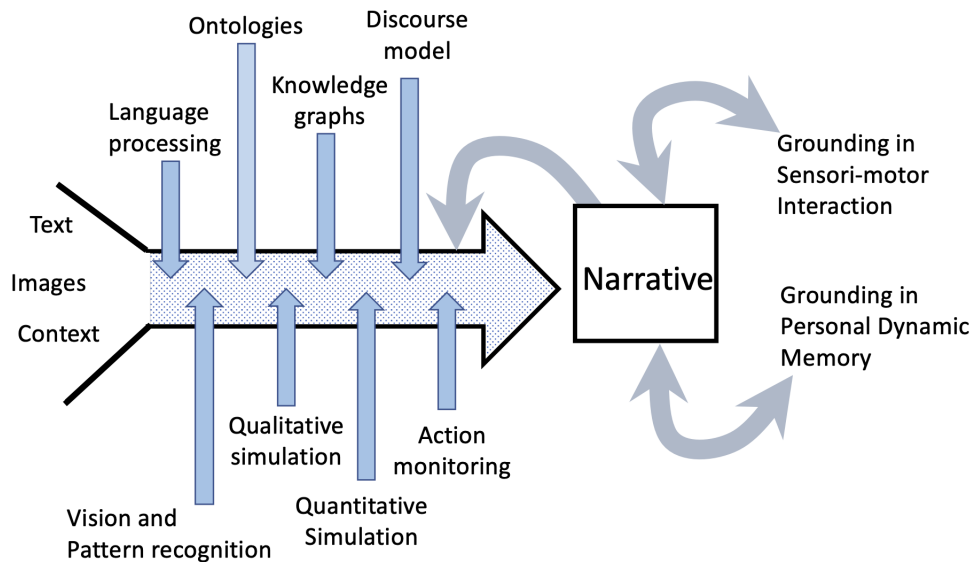


Figure 10.1: Narrative-based understanding is the process of reconstructing a rich content model underlying a text, image or situation, using a variety of knowledge sources and strategies for grounding the narrative in sensori-motor interaction and personal dynamic memory.

Close reading approaches a text, image or situation in terms of narratives. Narratives explicate the events, agents and their roles, the temporal, spatial and causal relations, the intentions and motivations of actions, as well as the perspectives and values implied by the actions and the situations described in the text [6]. Close reading is about reconstructing the narrative. This not only involves identifying the facts (the so-called *fabula* or world model) but also paying attention to how the facts have been organised and framed to fit with a specific viewpoint (the so-called *plot*) and deconstructing how both the *fabula* and *plot* have been communicated in a symbolic form [2]. The construction or interpretation of symbols is in itself often highly complicated, particularly in the case of language, requiring the construction or reconstruction of lexical, morphological, syntactic, semantic and pragmatic structures.

Close reading is the first step in understanding. The second step, which happens concurrently, is to ground the narrative in the real world as experienced through a sensorimotor embodiment (if the text describes real world actions or gives instructions for actions in the world) and in past experiences stored in personal dynamic episodic and semantic memory [25]. When this grounding is achieved we speak about *narrative-based understanding* (Figure 10.1).

Close reading has to go significantly beyond fine-grained perceptual or linguistic processing. Many additional sources of knowledge have to be brought to bear on the construction of the underlying narrative, simply because environmental cues, language, or images are ambiguous and vague and give only incomplete and fragmented information about the state of the world and how it is viewed by the narrator. These additional sources include:

- Information about the **context**, for example, the kitchen with its various cabinets containing utensils and food ingredients, a refrigerator, etc.
- **Ontologies**, that come into two different types: Conceptual ontologies that define categories with their components and relations in terms of frames. For example, a frame for a cooking pot will specify what information should be kept about the pot, such as what its size and contents are or whether the pot should be covered with a lid or not. Distributional ontologies that provide embeddings derived from the statistical processing of co-occurrences in texts and can be used for approximative matching.

- **Common sense and domain knowledge** contains large numbers of axioms that can be used to infer additional facts not explicitly mentioned in the text.
- **Discourse models** specify what is in the focus of attention and which questions are important and unresolved. For example, when the list of ingredients contains “three tomatoes”, these tomatoes come into the focus of attention. Hence, when the recipe later mentions “the tomatoes”, this description most likely refers back to these three tomatoes and not to other tomatoes that are still in the pantry.
- **Sensorimotor grounding** is about the relation between categories and sensory inputs or motor actions. This is relevant in situation awareness and image interpretation but also when a text contains images or is describing actions that are to be performed in a concrete context.
- **Qualitative or quantitative mental simulation** predicts either in a qualitative or quantitative way future or past world states. For example in a recipe the reader has to imagine the needs and effects of cooking actions because later steps will refer back to their outcome.

The grounding in the real world and in personal dynamic memory may also provide further feedback to constrain the possible interpretations coming out of close reading. Just like humans, narrative-based understanding systems need to use every possible bit of information and every possible knowledge source as quickly as possible in order to arrive at the most coherent model that integrates all data and constraints. This process cannot be implemented as a linear pipeline where one algorithm feeds into another, as is common in data-driven AI, because of a paradox known as the *hermeneutic circle* [12]: to understand the whole we need to understand the parts but to understand the parts we need to understand the whole [11].

Because of this paradox, narrative-based understanding has to be conceived of as a spiraling process. Starting from an initial examination of some input elements, for example a sentence with a lot of ambiguity, uncertainty and indeterminacy, hypotheses of the whole are constructed, which then provide top-down expectations to be tested by a more detailed examination of the same or additional sentences, leading to a clearer view of the whole, which then leads back to the examination of additional sentences, etc., until the narrative can be grounded in reality or episodic memory, reaching a state known as *narrative closure* [7].

In order to integrate all knowledge sources as quickly as possible, we should introduce a data structure that records progress in understanding. This paper presents such a data structure. It is called an **Integrative Narrative Network (INN)** and acts as a kind of blackboard on which different knowledge sources write partial descriptions and through which they can consult to advance aspects of understanding.

We operationalize the understanding process in terms of *narrative questions* and *answers*. The term “narrative question” is inspired by narratology where it is defined as what is happening in the audience’s mind (or what an author or screenwriter tries to make happen in their audience’s minds). Any particular moment in a narrative may raise new questions, and skilled writers know how to make their audience pose the desired questions. Narrative questions should therefore not be confused with questions as defined in Question-Answering systems, which are explicitly posed by humans; they are rather open slots in a narrative structure that can be filled in by different knowledge sources. In other words, each knowledge source can: (i) evoke new questions, (ii) introduce answers to questions, (iii) constrain the answers to questions, or (iv) shrink the set of questions by realizing that the answers to two different questions are in fact the same. It follows that the Integrative Narrative Network should take the form of a graph where the nodes are questions and answers and the links specify which answers relate to which

questions or which questions have the same answers. **Narrative closure** is reached when all critical questions have been answered, in particular the main questions raised by a text, image or situation.

10.2 Case Studies

The paper is illustrated with examples from two fully implemented case studies, one for recipe understanding and the other for visual dialogue. These case studies were developed and kindly shared by the Evolutionary & Hybrid AI research team at the VUB Artificial Intelligence Laboratory prior to the development of integrative narrative networks, demonstrating the general nature of our data structure.

10.2.1 The recipe understanding task

While narratives play an obvious role in literary-oriented AI fields such as story generation [19], we view narratives as a general cognitive tool for sense-making and reusing previous narratives to new situations [23], which means we can even apply narrative-based understanding to everyday activities such as cooking.

The recipe understanding task gets as input a recipe (text) and a simulated kitchen state, and then analyses the instructions to a point where the agent can do a mental simulations of the various actions that will be required to cook the recipe. Full detail is provided in a webdemo: <https://ehai.ai.vub.ac.be/demos/recipe-understanding/>

We use in this paper one of the recipes of the cooking benchmark, namely a recipe for preparing almond cookies. It goes as follows:

Recipe for almond cookies:

Ingredients: 226 grams butter, room temperature. 116 grams sugar. 4 grams vanilla extract, 4 grams almond extract, 340 grams flour, 112 grams almond flour, and 29 grams powdered sugar

Instructions:

1. Beat the butter and the sugar together until light and fluffy.
2. Add the vanilla and almond extracts and mix.
3. Add the flour and the almond flour.
4. Mix thoroughly.
5. Take generous tablespoons of the dough and roll it into a small ball, about an inch in diameter, and then shape it into a crescent shape.
6. Place onto a parchment paper lined baking sheet.
7. Bake at 175 degrees Celsius for 15 - 20 minutes.
8. Dust with powdered sugar.

The main narrative question posed by this recipe is how to prepare almond cookies, so narrative closure can be reached by successfully finishing the recipe. This question raises a host of other questions: What utensils are needed (a baking tray, a bowl, ...), where can things be found or put in the kitchen (freezer, pantry, ...), what ingredients are necessary (116 grams of sugar, 4 grams almond extract, ...), which objects need to be prepared (a mix of flour and almond flour, a small ball of dough, ...), which actions need to be performed (add flour, bake, ...), and properties of all these entities and actions. Understanding this and other recipes requires an integration of context, language (lexicon and grammar), a discourse model that spans across sentences and actions, qualitative simulation and ontologies. This paper cannot elaborate on the details for all these knowledge sources (see [ANONYMOUS] for a full description) but

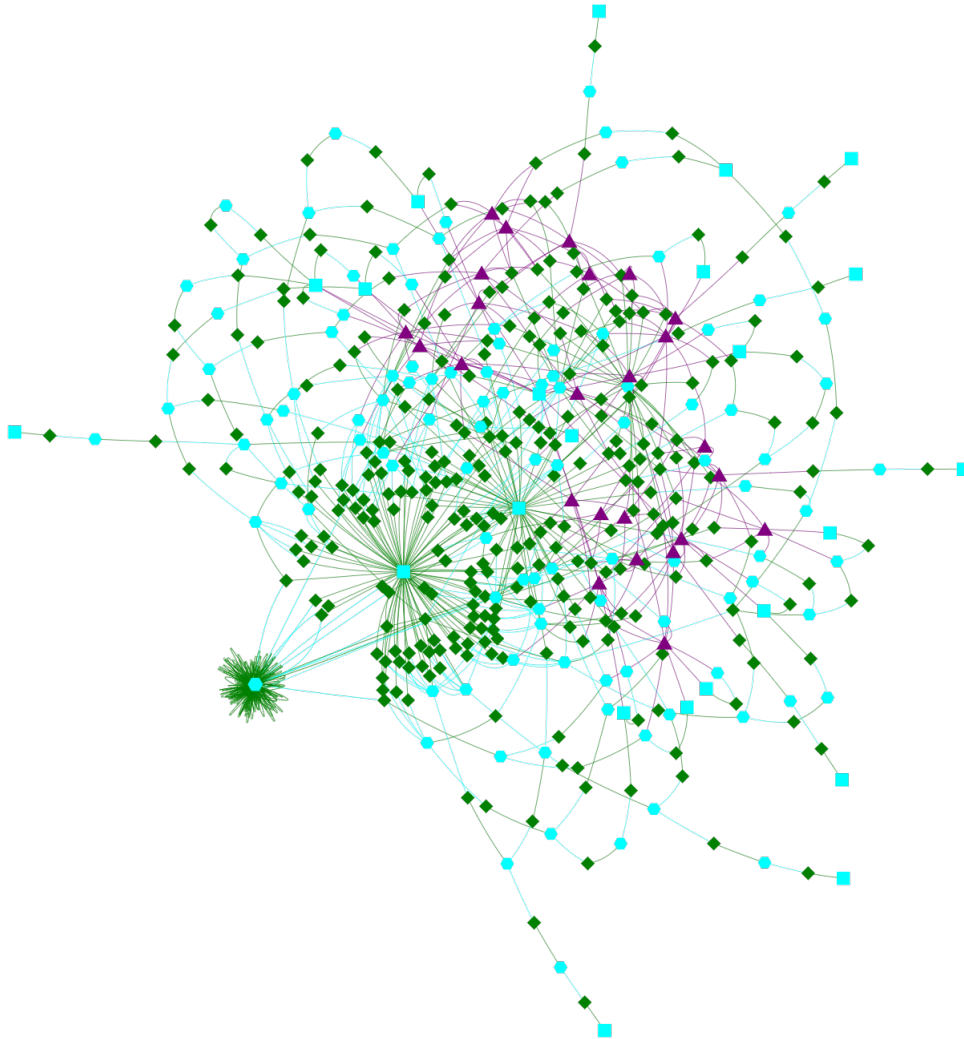


Figure 10.2: The Integrative Narrative Network after interpreting the Almond Cookies Recipe. Entities are displayed as cyan circles, linguistic information as purple triangles, and values as cyan squares. Narrative questions are visualized as diamonds, which here are all green which indicates that they have been successfully answered.

we will give a few concrete examples of processing systems later. The complete INN after processing this recipe is shown in Figure 10.2. The nature of this network and how it is built are explained further below.

10.2.2 The visual dialogue task

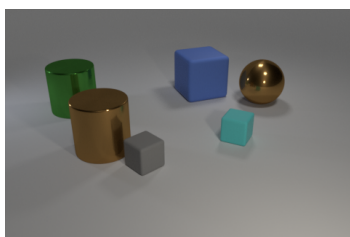
Visual dialogue is the task whereby an agent needs to hold a meaningful and coherent conversation with a human about visual input [8]. Concretely, dialogues from the CLEVR-Dialog dataset [14] have been used, which is a dataset with English language questions about CLEVR images [24]. An example of a dialogue is shown in Figure 11.1. For this case study, we use the implementation described in [45].

Just like the cooking domain, understanding linguistic dialogues about visual information requires a variety of knowledge sources: language processing for parsing the sentences into semantic representations, perception for finding the necessary information in the image that accompanies the dialogue, a discourse model for solving the co-references in the dialogues and inference operations applying common sense knowledge. In this case study, narrative questions are mainly invoked by language processing, while perception, discourse and inference typically provide the narrative answers. Figure 10.4 shows the INN after executing the dialogue shown in Figure 11.1.

10.2.3 Other case studies

We have done additional case studies using the same knowledge sources and including other ones, such as knowledge graphs and human interaction.

1. One case study is the close reading of a book about the French revolution. The text stimulates the reader to build models of the key actors, their intentions and motivations, the salient events, the temporal and causal relations between these events and the social and governmental changes they cause. The reader has to integrate these facts and viewpoints with her own personal dynamic memory containing narrative structures based on earlier readings or conversations and only if that successfully happens we can speak about understanding. Progress on this case study, particularly in relation to the inclusion of knowledge graphs for representing narrative structures, is reported in [3] and [32].
2. A second case study is about the interpretation of art works. Although most viewers spend only a few seconds on a painting in a museum – and hence have only a totally superficial experience because they engage only in distant reading – the force of a painting only appears on close reading, where the viewer carefully scans the painting, tries to figure out what the painting represents, what viewpoint and perspective has been adopted, how the painting is culturally embedded in a style, period, or political context, why this particular artist made that painting, etc. This case study, illustrated in Figure 10.5, is a human-in-control system in which a human user can search for an artwork from Wikidata.



Caption: A sphere is present in the scene.
 If there is an object in front of it, what color is it? - Green
 And material? - Metal
 How about the aforementioned round object? - Metal
 What is the shape of the above green object? - Cylinder
 Does the above round object have objects to its behind? - Yes
 What number of other objects in the picture share similar material with the above cylinder? - 2
 Any other objects? - Yes
 If there is an object in front of that cylinder, what color is it? - Cyan
 What is the size of the earlier cylinder? - Large
 What is the count of objects to its front in the picture? - 3

Figure 10.3: An example image from the CLEVR dataset, accompanied by a dialogue.

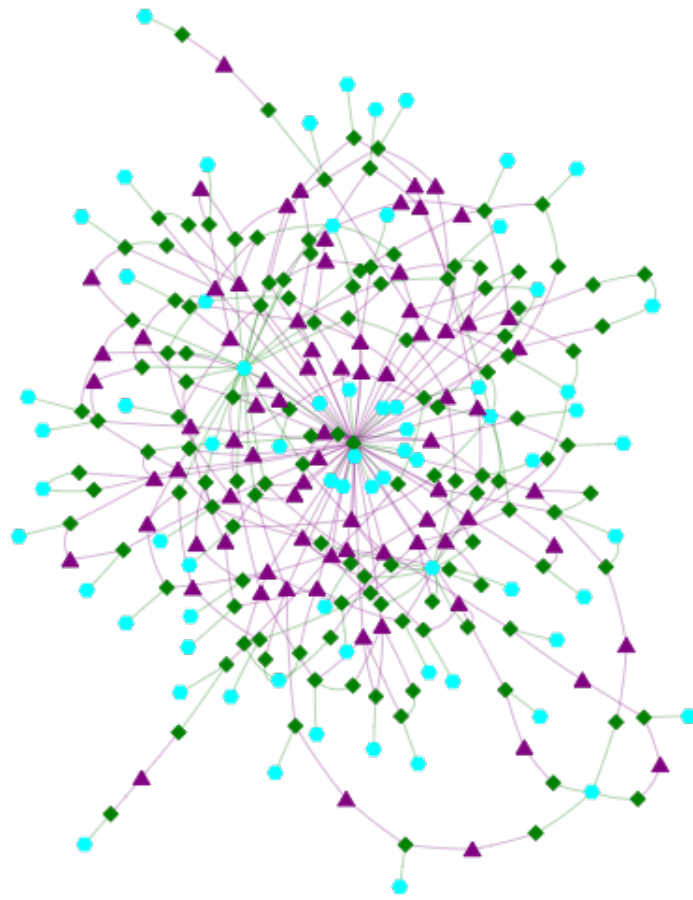


Figure 10.4: The Integrative Narrative Network after executing the dialogue shown in Figure 11.1. Given the size of the network only questions answered by inference through constraint propagation are shown.

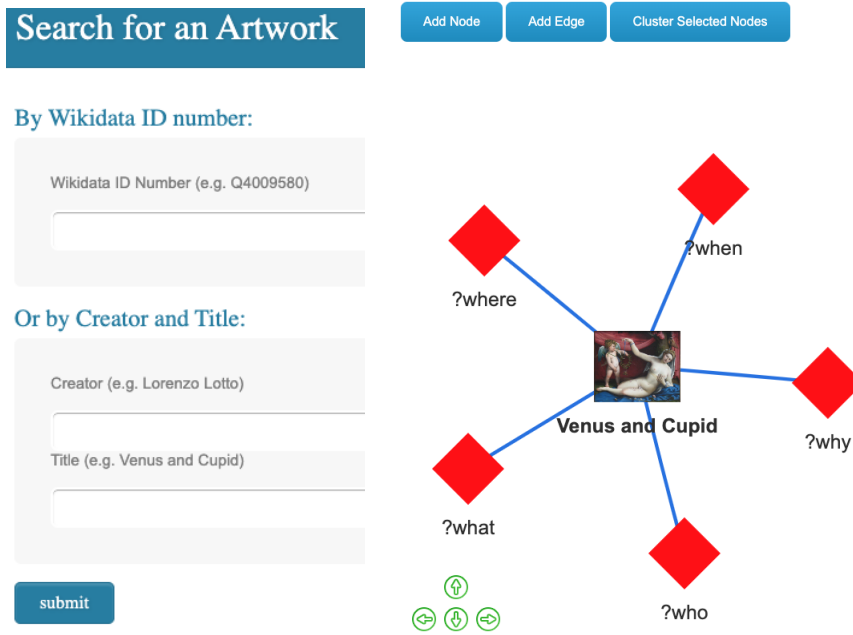


Figure 10.5: This Figure shows an ongoing, human-in-control case study on using integrative narrative networks for art exploration. The human user can start a search for an artwork on Wikidata, and interactively explore and extend the big WH-questions about the artwork.

The user can then explore five big “WH-questions” (who, what, why, where and when) for achieving a deeper understanding of the artwork. Besides steering the expansion of the network by clicking on open narrative questions (visualized as red diamonds), the user can add or delete nodes and edges. The art explorer case study integrates human interaction, vision, knowledge graphs (Wikidata) and an ontology of symbolic meanings [22, 21] for assisting the human user in the interpretation process. Progress on this case study, particularly the integration of machine vision, is reported in [1].

In the remainder of this paper only examples from the cooking domain and the visual dialogue task will be used.

10.3 Defining Integrative Narrative Networks

This section introduces the fundamental novel data structure introduced in this paper, namely the Integrative Narrative Network (INN). First we introduce the components of these networks and then how they are built by integrating the results of various knowledge sources.

10.3.1 Entities, Frames and Constraints

Entities are objects, events or (reified) concepts. They are designated with a symbol, called the entity-id and with angular brackets. The name is indicative of the type of entity being named and in this text we use subscripts to distinguish between different entities of the same type, as in $\langle \text{butter-331} \rangle$ or $\langle \text{bowl-710} \rangle$. Entities either refer to real world observational data (for example a region in an image), to virtual entities (which may or may not exist in reality), or to entities in a knowledge graph in which case we use the URI (Universal Resource Identifier) as unique identifier.

Entities may have different states, for example butter can be solid or become fluid when melted. To represent this, an entity has not only a persistent id but also different temporal existences,

each marked with its own identifier by the use of additional subscripts. For example, `<butter-331-1>` with the persistent id `<butter-331>` might change after heating into `<butter-331-2>` with the same persistent-id but different properties.

The handling of time is a complex and extensively-researched subject in qualitative physics research [9] and its discussion is beyond the scope of this paper. Nevertheless the implementations of the mental simulations for this paper use a temporal inference system with a time line and time proceeding as actions unfold. Entities come into existence at certain time steps but may also cease to exist. For example, the action of boiling water may take a certain amount of time and hence the pot with boiled water is only in existence at a future moment in time. Some actions discard entities (like bowls) after usage or entities may become an indivisible part of other entities, for example if butter and dough are being mixed.

Frames have a long tradition in AI knowledge representation and object-oriented programming, starting in the mid-seventies with the proposal by [16] and with technical realisations such as KRL [4], KRS [26] or CLOS [10]. A frame has a set of slots with values, which are in effect the questions that can be asked about a particular entity represented by a frame and the answers to these questions. An entity which is described with a certain frame is called a frame-instance of that frame.

Frames typically specify for each slot a default value, the type of the value, and possibly a procedure to compute the value (known as procedural attachment). Frames are organized in frame hierarchies with inheritance relationships between frames. Frames inherit slots and information about finding values from their parent-frames. The sequence of frames going from a specific frame (e.g. `linguine`) to its parent frames (e.g. `→ pasta → food ingredient → material object`) is called a frame lineage. The possible frames and their slots form the conceptual ontology of a domain.

When a frame lineage is used to describe a particular entity or set of entities, the lineage is first instantiated. In the remainder of this text frames and frame-instances are designated by symbols with square brackets. Names of instances have indices. In the recipe example, there is for example a frame for `[bowl]` with slots for the bowl itself (e.g. the contents, the size, the cover, whether the bowl has been used, etc). A specific bowl entity, e.g. `<bowl-75>`, is described by a frame instance, e.g. `[bowl-75]` and the name of the entity is stored in its persistent-id slot. In the current implementation, the world model is implemented in the style of object-oriented and frame-based knowledge representation systems and uses CLOS (the Common LISP Object System) [10] as implementation medium.

Constraints and Constraint Programming have an equally long tradition in AI knowledge representation [20], and we follow here in particular a tradition initiated by Sussman and Steele [30] with an implementation developed and used at the VUB Artificial Intelligence Laboratory and the Sony Computer Science Laboratory – Paris [27, 24]. IRL is itself implemented on top of CLOS.

A constraint is like a frame in the sense that it contains a set of slots, called arguments in constraint parlance. But rather than providing a procedure for computing the value of a single slot, a constraint specifies for combinations of known slot-values how other values can be computed.

A constraint performs computations to find answers to questions knowing the answers to others. Solving a constraint could involve creating new frame instances for additional entities, creating a new world state, copying an existing state with temporal changes, calling up other knowledge sources such as a quantitative simulation, consulting semantic web resources, etc. The constraint solver iterates through the constraints, progressively filling in as much information

as possible. Constraint solving is a search process because the same question may potentially have multiple answers until this is resolved by the application of more constraints. Moreover some questions may remain unanswered at the end of considering a constraint network, until more information arrives later.

A key point of the proposed design is that a constraint network does not rely on a fixed explicitly programmed control flow. The network only defines the data flow. In other words, there is no specific order in which information must be available to perform a computation and the constraint must be programmed to handle all possible ways in which information becomes available.

10.3.2 Narrative Questions and answers

We frame understanding as a process whereby the agent raises questions to itself, finds answers to questions, and interlinks questions and answers.

Questions are operationalized as variables. Following AI tradition, the name of a variable is written as a symbol with a question mark, as in ?variable-name, where the variable-name is a symbol that is chosen on the basis of the type of object that is an answer to the question. This is meant to be meaningful for us. Variable names are formal symbols that obtain their function only from their connections to other symbols not from the name they happen to have. Variable names typically have subscripts, as in ?bowl-1, ?bowl-2, ..., which refer to different bowls in the kitchen.

Answers The entities in the domain of discourse can all become answers to questions. The relation between a question and an answer is made operational in terms of a binding between variables and identifiers of entities or constants. A binding has a score reflecting the certainty with which the answer is estimated to be the right answer.

10.3.3 Visualization

The INN represents all questions that have been posed during the understanding process and all entities that appear as answers to these questions. The questions are visualized with green or red diamonds. The diamond is green if the question could be answered and red if it is still open. The answers to questions are represented with blue hexagons or squares.

A hexagon is used to represent an entity if the entity has an associated frame-instance. If the entity does not have an associated frame but is a constant, such as a number or a boolean value, it is represented as a square. Constraints are represented with purple triangles.

In addition, the graph represents three types of links:

1. The link between a question and the entity answering that question (in other words between a variable and the entity to which the variable is bound). This link is visualized using a green line.
2. The links between a constraint and the different variables (slots) of that constraint. They are visualized with purple lines.
3. The links between a frame instance and the different questions (slots) that the frame evokes. These are visualized with blue lines.

Two questions may have the same answer. For example, in the sentence “add water to the pot until it is full” The question to what the water must be added and the question what needs to be full are answered by the same entity, namely the pot. Often in the understanding process

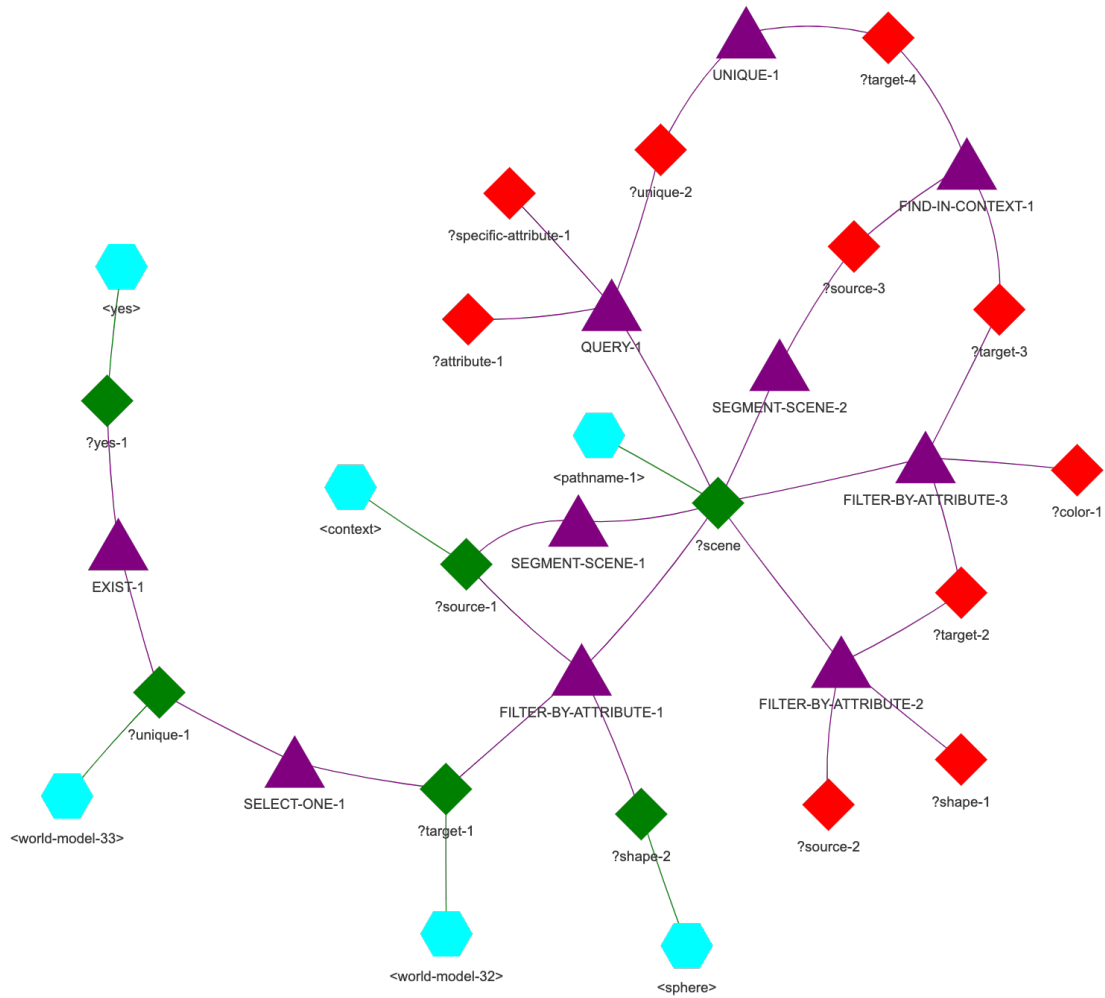


Figure 10.6: An example of an Integrative Narrative Network as part of the visual question-answering dialogue about geometric objects. Many questions remain unanswered.

these relationships (also known as co-referential links) only reveal themselves gradually. Our visualization tools have a parameter to either merge the co-referential question nodes to a single node or to leave them as separate and draw a link between the co-referential relations.

An example of an INN is shown in Figure 10.6. It represents a snapshot during the interpretation of a visual dialogue about geometric objects in a scene. Two sentences have been interpreted: “A sphere is present in the scene.” and “What is the shape of this cube?”. Note that these sentences are incoherent because it is not clear what “this cube” refers to as no cube has been mentioned yet. There are several red diamonds for these unanswered questions. We also see various purple triangles for the constraints, and a number of entities with frames (blue hexagons) which have generated more questions.

10.3.4 Formal definition of INN

Summarizing:

An *Integrative Narrative Network* (INN) I is a 6-tuple $I = \langle E, Q, C, F, A, N \rangle$ where

- E is the set of *entities*,
- Q is the set of *questions*,
- C is the set of *constraints*,
- F is the set of *frame lineages*,
- $N \subseteq C \times Q$, called the *constraint network*, is the set of relations between constraints and the questions they raise (i.e. the constraint arguments), and
- $A \subseteq F \times Q$ called the *answer graph*, is the set of relations between frames and questions they raise (i.e. the constraint slots).

Besides a formal specification, our paper is also supported by an open-source software implementation of integrative narrative networks (Apache 2.0 license) as part of the Babel cognitive software suite [15, 29] which can be downloaded at <https://emergent-languages.org/>.

10.4 Examples

To illustrate how Integrative Narrative Networks are built up, we now look at some snapshots during the understanding process drawing on the two case studies.

Contributions from the ontology to the INN The conceptual ontology defines the slots for a newly created entity based on the frame with which it is defined. Using an example from recipe understanding, consider the entity $\langle \text{butter-1} \rangle$ shown in Figure 10.7a. This entity is a frame-instance of the [butter] frame, and has the slots ‘melted’, ‘beaten’, ‘mixed’, ‘keep-refrigerated’. Each of these slots has a default value given in the ontology, either ‘t’ or ‘nil’. Thus, by introducing this entity, four questions are posed and are immediately answered.

The name of the variables for each of the slots are a combination of the name of the slot and the entity the slot belongs to: ?melted-butter-1, ?beaten-butter-1, ?mixed-butter-1, and ?keep-refrigerated-butter-1.

During qualitative mental simulation, entities are created and added to the Integrative Narrative Network (see Figure 10.7b). The entity itself is visualized by a blue hexagon node, connected with blue lines to the four questions raised by the frame associated with this entity. The questions are visualized as diamonds. They are colored green since they have been answered with default values (‘t’ or ‘nil’). The answers T and NIL are visualized with blue squares because there is no frame associated with these values that would itself generate still further questions. Default values can be overwritten later.

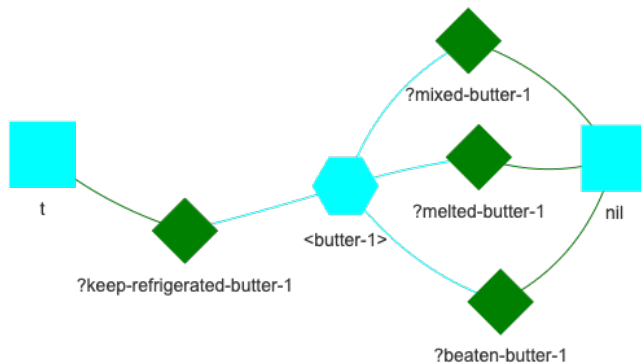
Contributions from language processing to the INN In our case studies, language processing uses precision construction grammars to parse utterances into rich linguistic structures (phrase structure, functional structure, case structure, information structure) and in the process extracting underlying frames and constraints to feed into the construction of the INN. In our case studies we rely on Fluid Construction Grammar for the parsing process [37, 44] and use semantic resources such as PROPBANK (<http://propbank.github.io/>, [18]) and various knowledge graphs to build the conceptual ontologies.

Consider parsing the utterance

“226 grams butter, room temperature.”

butter-1
melted: nil
beaten: nil
mixed: nil
keep-refrigerated: t
butter

(a) The <butter-1> entity, which is an instance of the [butter] frame with the slots ‘melted’, ‘beaten’, ‘mixed’ and ‘keep-refrigerated’.

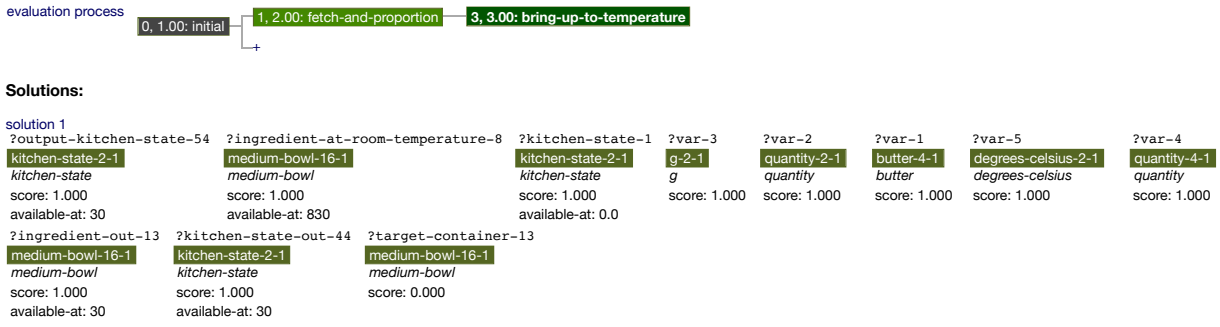


(b) The Integrative Narrative Network of the <butter-1> entity. Four questions are posed (?melted-butter-1, ?beaten-butter-1, ?mixed-butter-1, ?keep-refrigerated-butter-1) answered (so far) by their default values.

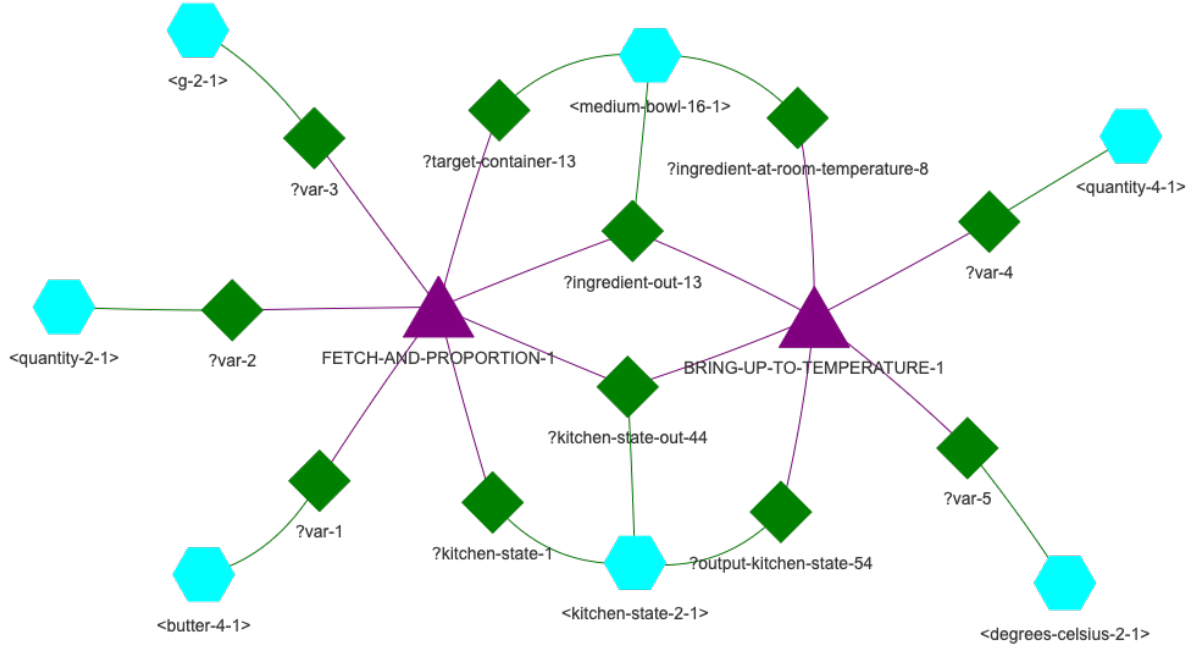
Figure 10.7: Contributions from the ontology to the INN

The *quantity-unit-ingredient-cxn* (shown in Figure 10.8a) looks for linguistic units with a numeral for the quantity, a noun for the unit and a noun for the ingredient. It also accesses the kitchen-state. When these linguistic units are found and the construction applies, it adds the constraint (FETCH-AND-PROPORTION ?ingredient-out ?kitchen-state-out ?kitchen-state-in ?target-container ?ingredient-in ?quantity ?unit) by instantiating it and introduces seven questions. Moreover, the construction also solves four questions that were previously introduced by linking the variables ?ingredient-in, ?quantity, ?unit and ?kitchen-state-in with variables from the linguistic units. This constraint and its arguments get instantiated and are added to the Integrative Narrative Network (see Figure 10.8a). The constraint FETCH-AND-PROPORTION-1 is represented as a triangle and introduces seven questions, represented as red diamonds. When these questions are answered later on, they will become green.

Contributions from qualitative mental simulation to the INN Qualitative mental simulation is implemented through the IRL constraint system. A constraint in IRL has associated methods that specify the bindings of certain variables given other variables and the state of the objects involved after the action has been undertaken. When these methods are invoked, the necessary entities are created or their properties are changed and the relevant variables are bound to these entities. For example, the constraint network in Figure 10.9a consists of two constraints FETCH-AND-PROPORTION and BRING-UP-TO-TEMPERATURE. The FETCH-AND-PROPORTION finds a certain ingredient of a certain quantity and unit in the kitchen state and proportions this ingredient and returns a new kitchen state with that ingredient. The constraint solving started from the initial bindings of <butter-4-1> to ?var-1, <quantity-2-1> to ?var-2, <g-2-1> to ?var-3 and <kitchen-state-2-1> to ?kitchen-state-1 provided in the language processing and bound the entity <medium-bowl-16-1> to ?ingredient-out-13, <medium-bowl-16-1> to ?target-container-13 and <kitchen-state-2-1> to ?kitchen-state-out-44, thereby providing 3 answers. Then, the BRING-UP-TO-TEMPERATURE constraint makes sure that this ingredient in the new kitchen state is brought up to room temperature and then returns the ingredient and another kitchen state. The constraint binds the entity <medium-bowl-16-1> to ?ingredient-at-room-temperature-8 and the entity <kitchen-state-2-1> to ?output-kitchen-state-54, thereby providing two answers. The bindings <quantity-4-1> to ?var-4 and <degrees-celsius-2-1> to ?var-5 were already provided during the language processing. The bindings <medium-bowl-16-



(a) Qualitative mental simulation solving the constraints in the network.



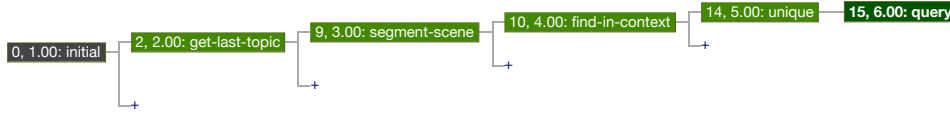
(b) The Integrative Narrative Network after the qualitative mental simulation. It consists of the constraints FETCH-AND-PROPORTION-1 and BRING-UP-TO-TEMPERATURE-1. These constraints pose different questions (the arguments of the constraints), which are all solved during the mental simulation (indicated by the blue nodes)

Figure 10.9: Contributions from qualitative mental simulation to the INN

1> to ?ingredient-out-13 and <kitchen-state-2-1> to ?kitchen-state-out-44 were made during the previous constraint. The constraint solving thus answers 5 questions. Then, the Integrative Narrative Network shows the two constraints in the form of purple triangles, linked to the arguments. All the arguments are bound to an entity, thus indicating that all questions are answered by the simulation.

Contributions from perception and discourse to the INN The knowledge sources of discourse and perception are consulted as part of constraint solving in IRL through constraints that call up procedures that consult the discourse model, trigger perception or perform inferences. Thus, in Figure 10.10a the constraint GET-LAST-TOPIC consults the discourse model to find the topic of the previous turn. The constraints SEGMENT-SCENE, FIND-IN-CONTEXT and QUERY are perception operations and respectively (i) segment the scene into the objects in the scene, (ii) find the previous topic in the scene and (iii) query the object for a shape. The UNIQUE constraint performs a logical operation to check whether there is only one object in its input.

evaluation process



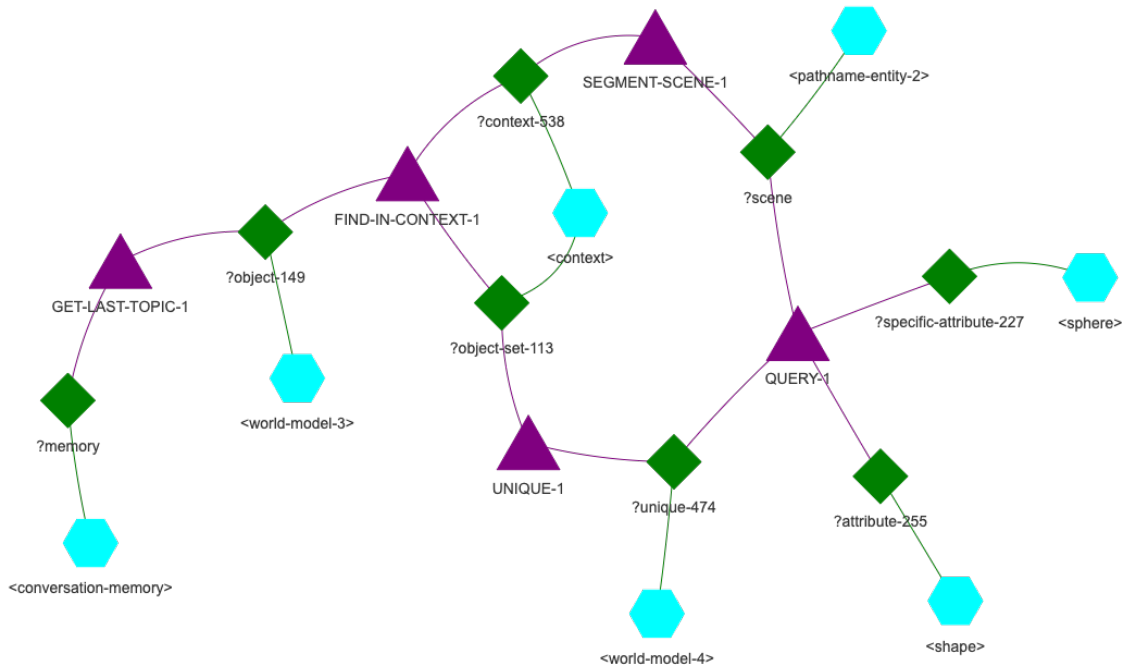
Solutions:

solution 1

?specific-attribute-227	?memory	?scene	?attribute-255	?object-149	?context-538	?object-set-113	?unique-474
sphere	conversation-memory	pathname-entity-2	shape	world-model-3	context	context	world-model-4
shape-category	world-model	pathname-entity	attribute-category	world-model	world-model	world-model	world-model
score: 1.000	score: 1.000	score: 1.000	score: 1.000	score: 1.000	score: 1.000	score: 1.000	score: 1.000

reset

(a) The execution of the constraint network as part of the visual dialogue example shown in Figure 11.1. The constraint GET-LAST-TOPIC consults the discourse model, the constraints FIND-IN-CONTEXT, SEGMENT-SCENE, QUERY consult the perception knowledge source and the UNIQUE constraint performs an inference operation.



(b) The INN after the execution of the constraint network. The constraints are indicated by purple triangles, their arguments by green nodes. The arguments of the constraints are solved by consulting the discourse model, performing perception, and inferencing. The answers are represented as blue hexagons or squares.

Figure 10.10: Contributions from discourse, perception and inferencing to the INN

These different knowledge sources thus contribute 5 answers, with 1 answer provided by the discourse model (the answer <world-model-3> for the question ?object-149), 3 answers provided by perception (the answer <context> for ?context-538 and ?object-set-113, and <sphere> for ?specific-attribute-227) and 1 answer provided by inference (the answer <world-model-4> for ?unique-474). The bindings of <conversation-memory> to ?memory, <pathname-entity-2> to ?scene and <shape> to ?attribute-225 were already solved by language processing.

The Integrative Narrative Network shown in Figure 10.10b shows the network after constraint solving. The five constraints are instantiated and visualised by purple triangles. The arguments that were posed and are visualized by green diamonds. Each of the questions is answered by an entity, represented with a blue square since these entities are not further associated with a frame here.

Dynamics of understanding We now look at an example that illustrates the dynamical process of understanding. Typically graphs are drawn after every step creating an animation of the understanding process. Figure 10.11 shows the INN at different moments of the execution of a small part of the almond cookies recipe introduced earlier, namely the two instructions

1. 226 grams butter, room temperature. 116 grams sugar.

followed by the instruction

2. Beat the butter and the sugar together until light and fluffy.

The first INN in Figure 10.11a shows the network during the qualitative simulation of “226 grams butter, room temperature.” The red nodes (unsolved questions) are connected to the relevant constraints. The qualitative simulation solves these open questions by binding their variables to entities. The second figure also shows the network during the qualitative simulation, with several remaining questions linked to constraints.

The third network shows the network right before narrative closure is reached, with only one remaining question. The last network is the INN at the end of the execution of the partial recipe, where all questions have been solved.

10.5 Using Integrative Narrative Networks

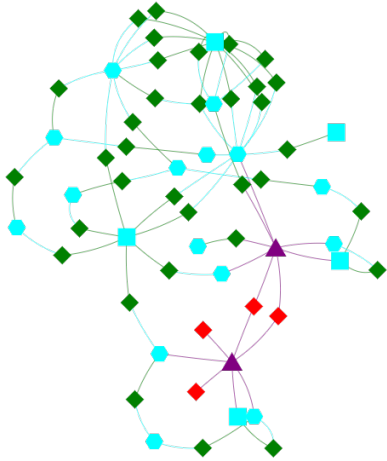
We now turn to the question why the INN is an important data structure for achieving and studying understanding processes. We have already seen its main function, namely to integrate information coming from a diverse set of knowledge sources. But there are additional functions:

- Measuring the contribution of different knowledge sources in the understanding process.
- Investigating quantitatively properties of a text such as narrative closure, coherence and efficiency.
- Achieve pro-active information gathering as part of understanding, including integrating human-in-the-loop interaction.
- Feedback from performance in understanding for machine learning.

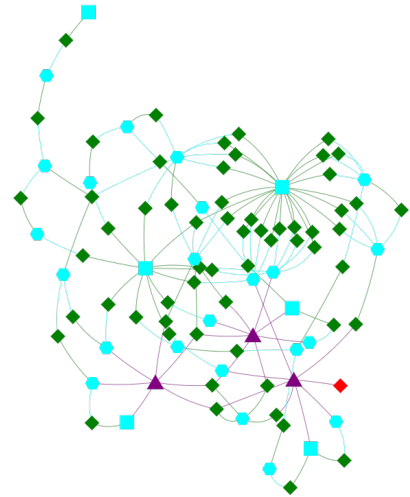
Each of these functions is now being discussed briefly.

10.5.1 Measuring the contributions of different knowledge sources

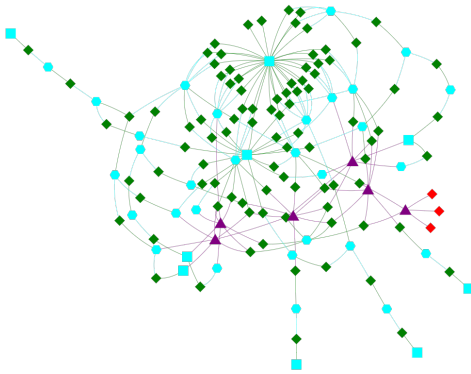
It is highly interesting to investigate the contribution of different knowledge sources in the understanding process. For example, one text may rely a lot on common sense knowledge and inference whereas another may rely heavily on an image next to the text. Identifying



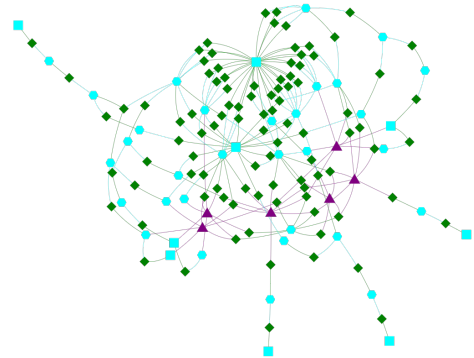
(a) The INN after parsing the first instruction “226 grams butter, room temperature. 116 grams sugar” and mentally simulating the fetch and proportion action.



(b) The INN after parsing the second instruction “Beat the butter and the sugar together until light and fluffy.” and mentally simulating the beating action.



(c) The INN with three remaining questions (indicated by red nodes).



(d) The INN at the end of executing a small recipe.

Figure 10.11: The INN at four different stages of the recipe execution.

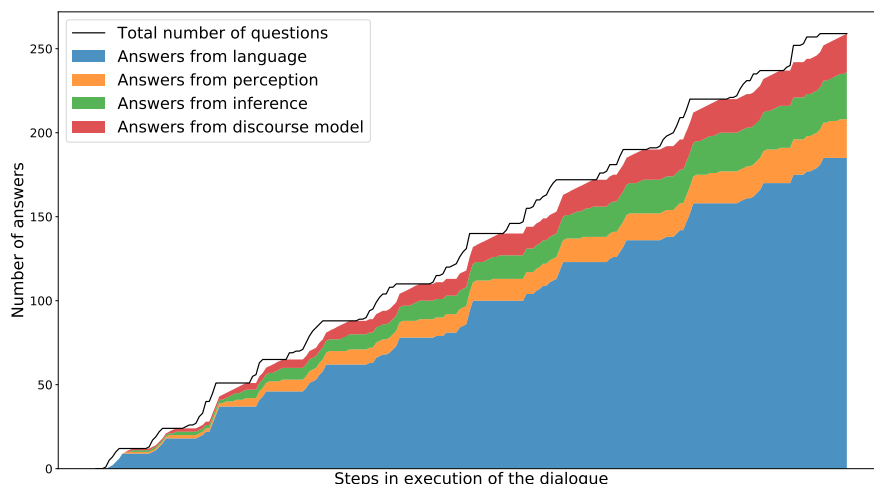


Figure 10.12: The contribution of the different knowledge sources when executing the visual dialogue shown in Figure 11.1. Notice that at each step (corresponding to a single input sentence) all questions raised so far are answered.

contributions is not only a function of the text, it also says something about the effectiveness of different knowledge sources. For example, an understanding system could be weak in language processing so that it has to rely more on other knowledge sources such as common sense inference, ontologies and world models. This is like a person who is less familiar with a foreign language but nevertheless tries to make sense of what is being said through the context and background knowledge.

To collect data during understanding we use a meta-level facility available in the BABEL architecture [29] used in the implementations reported. The architecture allows for the definition of monitors that become active when a triggering condition, for example the addition of a new node or a new link to the INN, is detected. The monitors then collect relevant information by observing the state of understanding at that point, including which knowledge source was responsible.

We have two concrete examples monitoring the increase in the number of questions that pop up during understanding and the increase in the number of answers that are found. Both the questions and the answers are coming from different knowledge sources and we monitor their contributions separately.

Figure 10.12 shows a first example, taken from the visual dialogue discussed earlier. Figure 10.13 shows a second example from processing the almond cookies recipe. In both examples we see that the number of questions steadily rises and questions get answered quickly after they have been raised. At the end of processing, all questions are answered. We also see that the contribution from language for the visual dialogue is much greater than in the cooking recipe, which relies much more on mental simulation and ontology.

10.6 Tracking narrative closure, coherence and efficiency

Not all texts are written well. This can be because the main questions are not clearly stated. The reader is left puzzled where the text leads to and hence becomes demotivated to read

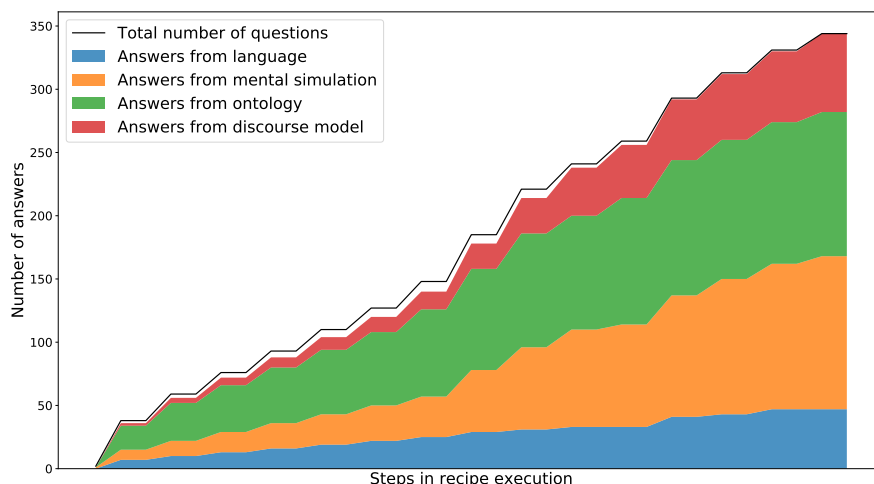


Figure 10.13: The contribution of the different knowledge sources during language analysis and mental simulation of the almond cookies recipe.

further. In addition, many texts introduce superfluous information which turns out to be irrelevant to answer the key questions of a text or they do not provide all the information that is relevant for the questions being posed. Often there is no straightforward way to find unexpressed information through common sense inference, visual perception, or mental simulation, because the author overestimates the reader.

But it can also be that the text is too verbose and introducing too much information explicitly that the reader already knows implicitly. It can also be that too many questions are left hanging after the interpretation of a sentence or paragraph. For example, a recipe may contain ingredients that are not used later or fails to mention ingredients in the beginning. These ingredients then suddenly pop up at a later stage and may require the cook to go fetch new ingredients or even start a preliminary cooking process that takes considerable amount of time. It may also be that critical operations are not explained and they fall outside the scope of common sense knowledge.

The INN makes it possible to develop quantitative measures for tracking these various issues in text quality and thus give feedback to authors:

(i) **Narrative closure** is computed by simply tracking the number of open questions during the interpretation process. It should ideally reach zero at the end of the text. If not, the INN can give back which questions remain unresolved. This measure can already be extracted from the graph shown earlier, where we see in the visual dialogue task that all questions being introduced are resolved before the next sentence is processed.

Here is an example (shown in Figure 10.14) that illustrates a case where narrative closure is not reached. It is for the domain of visual dialogue with the following sentences:

- A sphere is present in the scene.
- What is the color of this red cube?

In this dialogue, the second sentence refers to *this red cube*. However, no such object has been mentioned previously. Therefore, the referent of this phrase cannot be identified.

Figure 10.14 shows the INN at the end of the understanding process of this sub-optimal text.

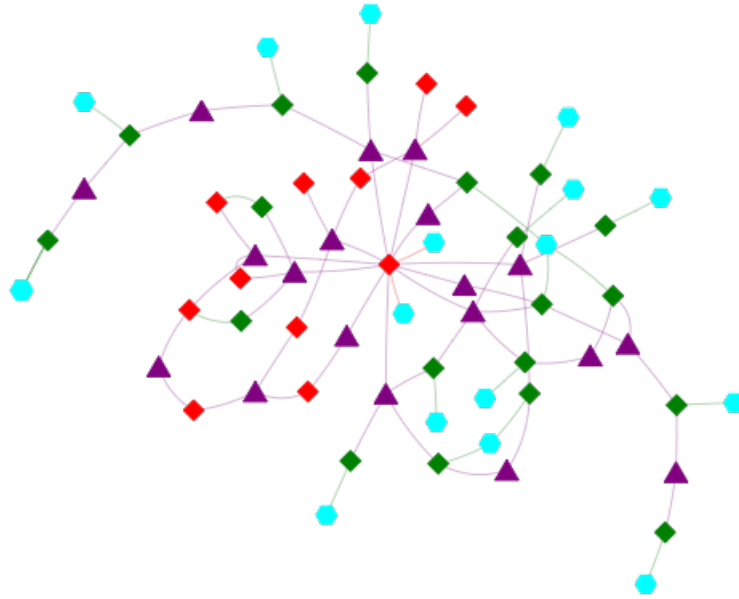


Figure 10.14: The narrative network when the system could not fully understand the dialogue as some questions are still remaining (indicated by red diamonds).

In this INN, some red diamonds are present, which are questions that the system could not resolve. Although language processing is able to parse the sentence “What is the color of this red cube?”, the consultation of the discourse model suggested by the word “this” cannot succeed because no earlier red object was mentioned.

(ii) **Narrative efficiency** measures the distance between the introduction of a question and the moment it gets answered. When the distance is larger, the interpreter has to remember the question longer and has to remain on the lookout for possible answers. The INN cannot only be used to measure narrative efficiency but also make suggestions about reordering parts of the text to make it more efficient for the reader.

(iii) **Narrative coherence** means that the INN is fully connected, in other words that the different sub-networks, which inevitably arise during interpretation, are linked with each other. An example of how coherence may improve during understanding is given in Figure 10.15 where we see how through language two disconnected INN fragments are coming together.

We now briefly discuss two additional functionalities which we have so far not yet worked out but are presented here to stimulate research in this direction.

10.6.1 Pro-active Understanding

Integrative Narrative Networks cannot only be used to observe the understanding process. Their structure, dynamics and quantitative characterizations can also be used to steer the understanding process itself, in particular it can guide the agent to actions that accelerate or improve understanding by focusing on certain questions and activating or giving more computational resources to knowledge sources that can help to resolve them.

Here are two examples:

1. When one of the knowledge sources is a vision system, this system can be called up actively to explore in more depth areas of the image that would help to resolve certain open questions that could not be handled through other means. This is certainly necessary

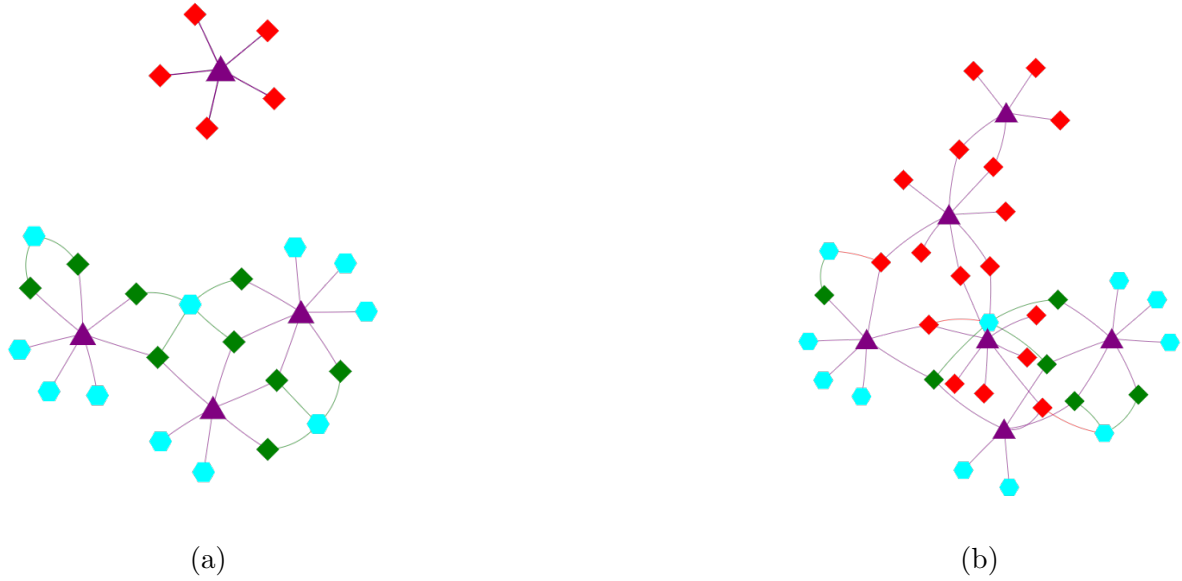


Figure 10.15: Language processing connecting the Integrative Narrative Network while processing the instruction “Beat the butter and the sugar together until light and fluffy”. One of the constraints is first disconnected from the network, but later connects to the rest of the network.

in the grounded cooking task, where objects and actions have to be linked to the perceived situation.

2. In human-machine dialogues, the understanding system can ask targeted help based on which questions in the INN are open at some point in time and could not be answered through other knowledge sources [31].

10.6.2 Feedback for machine learning

Finally, we just note that the various narrative measures that can be derived from Integrative Narrative Networks can play a role in machine learning, both for giving feedback on which knowledge sources are most effective (and therefore justify further training) but also in optimizing the overall understanding process in case there are scarce resources which should be allocated on the basis of where most progress can be made.

10.7 Conclusion

Understanding is the process of constructing a rich content model of a text, image or situation and ground this model in reality through a sensory-motor embodiment and in past experiences stored in a personal dynamic memory.

This paper contributes to the study and emulation of understanding by the introduction of a data structure called an Integrative Narrative Network (INN) that combines outcomes from diverse knowledge sources in a single graph. It views understanding as a process of posing questions to oneself, finding answers to these questions, and finding relations between different questions.

The paper defined INNs and illustrated its application drawing on fully implemented case studies for understanding recipes and engaging in visual dialogue. The paper also introduced the

potential for defining various measures over INNs that are helpful for quantitatively analyzing the understanding process.

The work presented here is just one step in building a quantitative infrastructure for tracking and evaluating understanding in AI systems. Having quantitative measures is useful to pin down precisely the contribution of a particular knowledge source or to provide feedback to the attention mechanism that guides what knowledge sources should preferentially be used or what questions should be the focus of attention. Quantitative measures can also play a role as feedback signal for improving the efficiency and efficacy of understanding.

Chapter 10: References

- [DB1] Sinem Aslan and Luc Steels. Aligning figurative paintings with their sources for semantic interpretation. In Press, 2023.
- [DB2] M. Bal and C. Van Boheemen. *Narratology: Introduction to the theory of narrative*. University of Toronto Press, Toronto, Canada, 1997.
- [DB3] Inès Blin. Building a French revolution narrative from wikidata. In Lise Stork, Katrien Beuls, and Luc Steels, editors, *Proceedings of the Workshop on Semantic Techniques for Narrative-Based Understanding co-located with 31st International Joint Conference on Artificial Intelligence and the 25th European Conference on Artificial Intelligence (IJCAI-ECAI 2022)*, pages 1–5, Vienna, 2022. CEUR Workshop Proceedings.
- [DB4] Daniel G. Bobrow and Terry Winograd. An overview of KRL, a knowledge representation language. *Cognitive Science*, 1(1):3–46, 1977.
- [DB5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, Red Hook, NY, USA, 2020. Curran Associates, Inc.
- [DB6] J. Bruner. The narrative construction of reality. *Critical Inquiry*, 18(1):1–21, 1991.
- [DB7] N. Carroll. Narrative closure. *Philosophical Studies.*, 135:1–15, 2007.
- [DB8] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh, and Dhruv Batra. Visual Dialog. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1080–1089, Los Alamitos, CA, USA, 2017. IEEE Computer Society.
- [DB9] K. Forbus. *Qualitative Physics: Past, Present and Future.*, pages 239–296. Exploring Artificial intelligence. Morgan Kaufman, 1988.
- [DB10] Richard P. Gabriel, Jon L. White, and Daniel G. Bobrow. CLOS: Integrating object-oriented and functional programming. *Commun. ACM*, 34:29–38, 09 1991.
- [DB11] H-G.. Gadamer. Hermeneutics and social science. *Cultural Hermeneutics*, 2(4):207–316, 1975.
- [DB12] Martin Heidegger. *Being and Time*. SCM Press, London, 1962. Transl. by John Macquarrie and Edward Robinson.
- [DB13] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. CLEVR. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2901–2910. IEEE Computer Society, 2017.

- [DB14] Satwik Kottur, José M. F. Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. CLEVR-Dialog: A diagnostic dataset for multi-round reasoning in visual dialog. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 582–595, Minneapolis, MN, USA, 6 2019. Association for Computational Linguistics.
- [DB15] Martin Loetzsch, Pieter Wellens, Joachim De Beule, Joris Bleys, and Remi van Trijp. The Babel2 Manual. Technical Report AI-Memo 01-08, AI-Lab VUB, Brussels, 2008.
- [DB16] Marvin Minsky. A framework for representing knowledge. In P. Winston, editor, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill, New York, 1975.
- [DB17] F. Moretti. Conjectures on world literature. *New Left Review*, 1, 2000.
- [DB18] Martha Palmer, Dan Gildea, and Paul Kingsbury. The Proposition Bank: A corpus annotated with semantic roles. *Computational Linguistics*, 31(1):71–106, 2005.
- [DB19] Mark O. Riedl and R. Michael Young. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39:217–268, 2010.
- [DB20] Francesca Rossi, Peter van Beek, and Toby Walsh, editors. *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*. Elsevier, 2006.
- [DB21] Bruno Sartini, Sofia Baroncini, Marieke van Erp, Francesca Tomasi, and Aldo Gangemi. Icon: an ontology for comprehensive artistic interpretations. *Journal on Computing and Cultural Heritage*, 2023. accepted for publication.
- [DB22] Bruno Sartini, Marieke van Erp, and Aldo Gangemi. Marriage is a peach and a chalice: Modelling cultural symbolism on the semantic web. In *K-CAP ’21: Proceedings of the 11th on Knowledge Capture Conference*, pages 201–208, New York, 2021. ACM.
- [DB23] Roger C. Schank. *Tell Me a Story: Narrative and Intelligence*. Northwestern University Press, Evanston IL, 1995.
- [DB24] Michael Spranger, Simon Pauw, Martin Loetzsch, and Luc Steels. Open-ended procedural semantics. In Luc Steels and Manfred Hild, editors, *Language Grounding in Robots*, pages 153–172. Springer, New York, NY, USA, 2012.
- [DB25] L Steels. Conceptual foundations for human-centric AI. In M. Chetouani, V. Dignum, P. Lukowicz, and C. Sierra, editors, *Advanced course on Human-Centered AI. ACAI 2021*, volume LNAI Tutorial Lecture Series. Springer Verlag, 2022.
- [DB26] Luc Steels. Design requirements for knowledge representation systems. In Joachim Laubsch, editor, *GWAI-84*, pages 1–19, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.
- [DB27] Luc Steels. The emergence of grammar in communicating autonomous robotic agents. In *ECAI 2000: Proceedings of the 14th European Conference on Artificial Intelligence*, pages 764–769, Amsterdam, 2000. IOS Press.
- [DB28] Luc Steels, editor. *Design patterns in Fluid Construction Grammar*. John Benjamins, Amsterdam, Netherlands, 2011.
- [DB29] Luc Steels and Martin Loetzsch. Babel: A tool for running experiments on the evolution of language. In Stefano Nolfi and Marco Mirolli, editors, *Evolution of Communication and Language in Embodied Agents*, pages 307–313. Springer, Berlin, Germany, 2010.

- [DB30] Gerald Jay Sussman and Guy Lewis Jr Steele. Constraints—a language for expressing almost-hierarchical descriptions. *Artificial Intelligence*, 14(1):1–39, 1980.
- [DB31] Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Nick Walker, Yuqian Jiang, Harel Yedidsion, Justin Hart, Peter Stone, and Raymond J. Mooney. Jointly improving parsing and perception for natural language commands through human-robot dialog. *Journal of Artificial Intelligence Research*, 67:327–374, 2020.
- [DB32] R. van Trijp and I. Blin. Narratives in historical sciences. In L. Steels, editor, *Foundations for Incorporating Meaning and Understanding in Human-centric AI*. “MUHAI consortium”, 2022.
- [DB33] Remi van Trijp, Katrien Beuls, and Paul Van Eecke. The FCG Editor. *PLOS ONE*, 17(6):e0269708, 2022.
- [DB34] Lara Verheyen, Jérôme Botoko Ekila, Jens Nevens, Paul Van Eecke, and Katrien Beuls. Hybrid procedural semantics for visual dialogue. In *Workshop on semantic techniques for narrative-based understanding: Workshop at IJCAI-ECAI 2022*, pages 48–52, 2022.

Chapter 11

Learning large-scale construction grammars¹

Jonas Doumen^{†,◇} Katrien Beuls[△], Paul Van Eecke^{*}

[†] Research Unit Linguistics, Faculty of Arts, KU Leuven

[◇] imec research group itec, KU Leuven

[△]Faculté d’Informatique, Université de Namur

^{*}Artificial Intelligence Laboratory, Vrije Universiteit Brussel

Abstract

Recently, substantial breakthroughs have been achieved in the automatic learning of computational construction grammars through methodologies that are inspired by the processes underlying child language acquisition. With these methodologies, computationally tractable, large-scale, usage-based grammars can be learned in an online, interactive, incremental, transparent and data-efficient manner. By exhibiting such human-like properties, these methodologies form an excellent basis for modeling the language acquisition abilities in human-centric and meaningful AI systems. While these methodologies can be used for learning computational constructions grammars for the cooking domain from recipe texts, here we report on preliminary work where they are applied to template-based natural language questions.

11.1 Introduction

Usage-based theories of language acquisition argue that the ability of children to learn language is based on two general cognitive capacities: *intention reading* and *pattern finding* [38, 39]. Intention reading refers to the capacity of children to understand the communicative intentions of their interlocutors. Pattern finding refers to the ability to recognise similarities and differences in sensory-motor experiences, and to use this ability for categorisation and schema formation [38, p. 3–4]. Pattern finding thus provides mechanisms for generalising across different communicative interactions, thereby constructing abstract schemata that represent the linguistic knowledge of a language user. In the context of language acquisition, intention reading and pattern finding are two key cognitive capacities that are highly complementary. Intention reading

¹This work was previously published as: Doumen, J., Beuls, K., and Van Eecke, P. (2023). Modelling language acquisition through syntactico-semantic pattern finding. In Vlachos, A. and Augenstein, I., editors, *Findings of the Association for Computational Linguistics*, pages 1317– 1327. Association for Computational Linguistics.

allows a language learner to reconstruct the meaning of an utterance that they observe during a communicative interaction. Pattern finding then provides the mechanisms to learn a grammar based on the combination of these observed utterances and their reconstructed meanings.

There exists an impressive body of theoretical and empirical evidence for both intention reading [6, 32, 26, 28] and pattern finding [22, 9, 11, 23]. However, no comprehensive mechanistic models that provide a faithful operationalisation of either of these cognitive processes exist to date. Here, we aim to fill part of this void by presenting a computational operationalisation of pattern finding mechanisms that can bootstrap a grammar based on a set of semantically annotated utterances alone. As such, we assume that the outcome of the intention reading process is given, hence the availability of the utterances’ semantic representations, but that neither a segmentation of the utterances nor any pre-existing morpho-syntactic or other grammatical information can be used. For a computational model that operationalises the intention reading process, and that integrates it with the pattern finding mechanisms introduced below, we refer the interested reader to [29].

A validation of our methodology on the CLEVR benchmark dataset for visual question answering [24] shows that it allows for fast, incremental and effective grammar learning. The result of this learning process is a fully-operational, productive construction grammar that can be used for both language comprehension, i.e. mapping from utterances to their meaning representation, and language production, i.e. mapping from a meaning representation to an utterance.

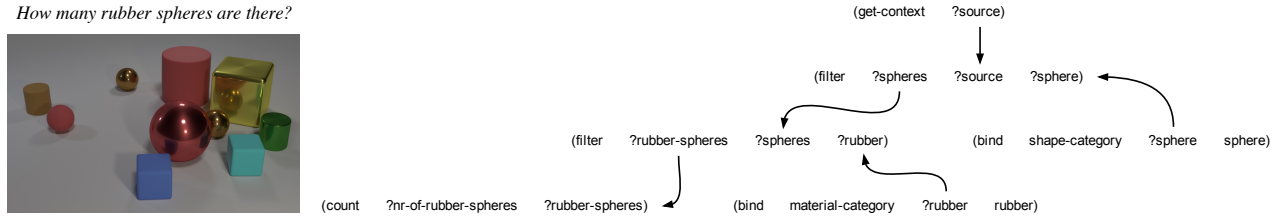


Figure 11.1: Example scene, question, and procedural semantic representation from the CLEVR dataset.

The scientific contribution of our work is twofold. On the one hand, it provides computational evidence for the cognitive plausibility of usage-based theories of language acquisition by introducing a mechanistic model of the acquisition of construction grammars from scratch. On the other hand, the techniques that we present pave the way for learning computationally tractable, large-scale, usage-based construction grammars that facilitate both language comprehension and production. Apart from their theoretical importance, such grammars are also highly valuable for a large range of application domains, including intelligent conversational agents [45] and the semantic analysis of discourse [46, 5].

The remainder of this text is structured as follows. Section 11.2 presents the dataset, task and learning problem that we address. Section 11.3 introduces our novel methodology for learning construction grammars. Section 11.4 presents the evaluation results. Related work is discussed in Section 11.5. A concluding discussion is provided in Section 11.6.

11.2 Data

There are two main requirements for datasets to be compatible with the methodology that we present here. First of all, they need to consist of utterances that are annotated with a representation of their meaning. Second, they need to be large enough so that they contain

enough utterances that are similar to each other, but not equal, in terms of either form or meaning. The availability of exemplars that are sufficiently close to each other is a necessary precondition for any generalisation process and is fully consistent with the prevailing hypotheses of how children acquire language [38]. The exact required size of a dataset is as a consequence directly related to the variety and the degree of complexity of the utterances and meaning representations that it contains.

We present and validate our methodology using the CLEVR dataset for visual question answering [24]. The utterances in the dataset are semantically annotated and the dataset contains ample examples of utterance-meaning pairs that are similar but not equal to each other. The utterances are English questions about images of scenes depicting different configurations of geometrical figures. Each question is annotated with a semantic representation that captures the logical meaning that underlies it. An example of such a scene, a question and its semantic representation is shown in Figure 11.1.

The semantic representation in Figure 11.1 takes the form of a set of predicates that share arguments with each other. In the figure, the predicates are drawn in the form of a network, based on the variables that they share. The meaning representation of a question can naturally be represented as a query, i.e. a series of steps that need to be taken in order to answer the question. Each predicate represents a step in this reasoning process, and intuitively corresponds to an atomic cognitive operation that a human or machine can perform. In the case of the example utterance ‘*How many rubber spheres are there?*’, the reasoning process consists of four main steps. The first predicate, GET-CONTEXT, binds the image to the variable ‘?source’. Then, the FILTER predicate filters the image for instantiations of the concept of SPHERE. The result of this filtering operation, i.e. the set of all spheres that are in the image, is bound to the variable ‘?spheres’. This set of spheres is subsequently filtered by another FILTER predicate for instantiations of the concept of RUBBER. The resulting set of rubber spheres is bound to the variable ‘?rubber-spheres’. Finally, the set of rubber spheres is counted by the COUNT predicate and the result is bound to the variable ‘?nr-of-rubber-spheres’. The meaning of the question ‘*How many rubber spheres are there?*’ corresponds thus informally to filtering an image for spheres, filtering the spheres for rubber objects and counting the result of this last operation. Such meaning representations are called *procedural semantic representations* as the representations themselves are at the same time executable procedures [47, 25]. Our methodology handles procedural semantic representations without problems, but is in no way restricted to it. It can handle any semantic representation, as long as it embraces some notion of compositionality and can be expressed as a set of predicates. Examples of other compatible semantic representations include abstract meaning representation [3], PropBank frames [30] and the lambda calculus [8, 27].

The CLEVR dataset consists of three splits: a training split of 70,000 images and 699,989 questions, a validation split of 15,000 images and 149,991 questions, and a test split of 15,000 images and 149,988 questions. The questions in the training and validation splits come with semantic annotations, whereas the test set does not. As we require these annotations in order to evaluate our model, we use the training split of the CLEVR dataset as training set and the validation split as test set. The question-annotation pairs embrace various aspects of reasoning, including attribute identification (‘*There is a large cube; what is its color?*’), counting (‘*How many green spheres are there?*’), comparison (‘*Are there an equal number of large cubes and small things?*’), spatial relationships (‘*What size is the cylinder that is right of the yellow shiny thing that is left of the cube?*’) and logical operations (‘*How many objects are either red cubes or yellow cylinders?*’). For the purposes of this work, we have selected the subset of CLEVR questions that do not involve comparison, spatial relationships or logical operations. The main reason for this is that these are complex cognitive operations that often correspond to long and

complex utterances that are far removed from the linguistic expressions that children (or even other humans) are faced with. Our final training and test sets consist of 47,134 questions and 10,044 questions respectively.

The learning task that we address consists in operationalising pattern finding mechanisms that facilitate the learning of a bidirectional construction grammar. The grammar should be able to map between the CLEVR utterances and their semantic representation, both in the comprehension (form to meaning representation) and the production (meaning representation to form) direction.

11.3 Methodology

The input to the learning process consists of utterances that are annotated with a representation of their meaning. The output of the learning process should consist in form-meaning mappings (constructions) that can be used for comprehending and producing utterances. The form-meaning mappings are represented in, and processed using, Fluid Construction Grammar [37, 41, 44].

11.3.1 Holophrase Constructions

Let us for a moment take the perspective of the learning algorithm. At the beginning of the learning process, the construction inventory is empty and the first utterance-meaning pair from the corpus comes in. At this point, the only thing that the learning algorithm can do is to store an exact mapping between the observed form and its meaning. Such a holistic mapping corresponds to a holophrase construction and is usable as such, albeit only for comprehending and producing the exact same utterance as the one that was observed. In order to use such a construction in the comprehension direction, it suffices to match the form side of the construction with an utterance and return the meaning side of the construction if the matching process succeeded. In order to use the same construction in the production direction, the meaning side of the construction must be matched with a semantic network and the form side must be returned.

When a next observation comes in, the learning algorithm first checks whether it is already covered by constructions that have been acquired previously. When this is the case, the constructions that are involved in the successful comprehension and production of the observation are reinforced by incrementing their entrenchment score. If the observation is not covered, the algorithm checks whether there are any generalisations that can be made based on the combination of the novel observation and any previously acquired constructions. It is these generalisation mechanisms that embody Tomasello’s pattern finding capacity [38] and are thereby at the core of the construction learning process. We have identified three classes of mechanisms that facilitate the learning of general constructions by algorithmically reasoning over similarities and differences between existing constructions and novel observations.

11.3.2 Generalising over Holophrase Constructions

The first class of mechanisms facilitates the generalisation of holophrase constructions with respect to novel observations. These mechanisms can learn item-based constructions that capture the similarities between a novel observation and an existing holophrase construction that was learnt based on a similar, but not equal, observation. These item-based constructions abstract away from the differences between the observation and the holophrase construction.

For example, imagine that a holophrase construction has already been learnt based on the observation of the utterance ‘*How many rubber spheres are there?*’ and the semantic network shown in Figure 11.1. Now, a novel utterance ‘*How many rubber cubes are there?*’ is observed, along with a very similar meaning network in which the predicate ‘(bind shape-category ?cube cube)’ appears at the place of ‘(bind shape-category ?sphere sphere)’. The generalisation mechanisms compute the similarities and differences between the construction and the observation in terms of both form and meaning, and make a new item-based construction that maps between the utterance ‘*How many rubber ?X are there?*’ and the semantic network from Figure 11.1 in which the non-overlapping predicate has been replaced by a variable. At the same time, two new lexical constructions are created, which capture the differences between the observation and the original holophrase construction. In our example, these will be a construction that maps between the utterance ‘*cubes*’ and the meaning representation ‘(bind shape-category ?cube cube)’ and a construction that maps between the utterance ‘*spheres*’ and the meaning representation ‘(bind shape-category ?sphere sphere)’. Finally, categorial links are made between the ?X slot in the item-based construction and the new lexical constructions. These categorial links capture that ‘*cubes*’ and ‘*spheres*’ can both appear in the ?X slot of the construction for ‘*How many rubber ?X are there?*’. A schematic representation of this learning process is shown in Figure 11.2.

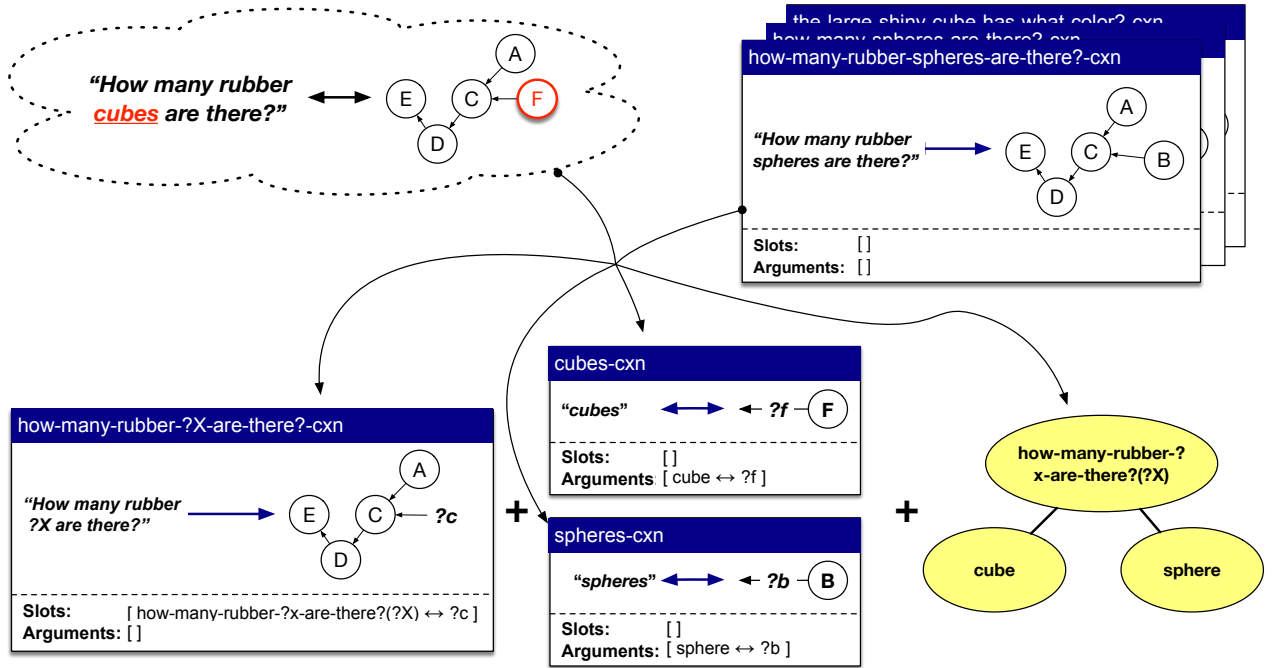


Figure 11.2: A schematic representation of a generalisation operator learning an item-based construction and two lexical constructions based on an existing holophrase construction and a novel observation.

There are three different scenarios in which mechanisms of this class are active. The first scenario concerns utterances which extend holophrases that are already known. An example would be the generalisation of ‘*Are there any cylinders?*’ to ‘*Are there any red cylinders?*’. In this case, an item-based construction ‘*Are there any ?X cylinders?*’ is learnt, along with a lexical construction for ‘*red*’ and a categorial link between the lexical construction and the open slot in the item-based construction. The second scenario concerns utterances which reduce known holophrases. An example would be the reduction of ‘*What is the size of the metal block?*’ to ‘*What is the size of the block?*’. In this case, an item-based construction for ‘*What is the size of the ?X block?*’ is learnt, along with a holophrase construction for ‘*What is the size of the block?*’, a lexical construction for ‘*metal*’, and a categorial link between the slot in the

item-based construction and the lexical construction. The final scenario concerns utterances which are not a mere extension or reduction of each other, but contain different formal and/or semantic material. An example would be the utterances ‘*How many rubber spheres are there?*’ and ‘*How many rubber cubes are there?*’ discussed above, where a holophrase construction for ‘*How many rubber spheres are there?*’ is already in place. An item-based construction for ‘*How many rubber ?X are there?*’ is learnt along with a lexical construction for ‘*cubes*’ and a categorial link between the open slot in the item-based construction and the new lexical construction. Additionally, a second lexical construction for ‘*spheres*’ is learnt, along with a categorial link between the open slot in the item-based construction and the lexical construction for ‘*spheres*’.

11.3.3 Learning Constructions Based on a Partial Analysis

The second class of mechanisms is designed to handle cases where an observation could not completely be processed using the existing constructions of a grammar, but where a partial analysis could be provided. These mechanisms can then create novel constructions that can work together with existing constructions so that the entire observation can be processed successfully. They start thus from the combination of a novel observation on the one hand, and an item-based construction or one or more lexical constructions on the other. The second class of mechanisms is active in two different scenarios.

The first scenario concerns observations to which an item-based construction can apply, but where there remains material that is not covered by any of the existing constructions. An example would be an observation of ‘*What is the size of the green block?*’, where a construction for ‘*What is the size of the ?X block?*’ is already known, while no construction for ‘*green*’ has been learnt yet. The learning algorithm detects that some aspects of the form and the meaning of the observation are not covered by the existing item-based construction and it creates a novel lexical construction that maps between those parts of the form and meaning that were not covered. Additionally, a categorial link is made between the slot in the item-based construction and the lexical construction. In our example, this means that a lexical construction for ‘*green*’ is learnt, along with a categorial link between this construction and the ?X slot in the construction for ‘*What is the size of the ?X block?*’.

The second scenario concerns observations to which one or more lexical constructions can apply, but where these constructions do not fully cover the input. An example would be an observation of the utterance ‘*There is a big red cube; what is its material?*’, where lexical constructions for ‘*big*’, ‘*red*’, ‘*cube*’ and ‘*material*’ have already been learnt. The learning algorithm will then create a new item-based construction that incorporates all the form and meaning material that remains after the application of these lexical constructions, and that abstracts away from these constructions through the integration of four slots. The result is an item-based construction of the form ‘*There is a ?A ?B ?C; what is its ?D?*’ and four categorial links from the existing lexical constructions to the slots in the new item-based construction.

11.3.4 Extending the Categorial Network

The third class of mechanisms is designed to handle cases where all necessary constructions are already in place, but where they cannot combine due to the absence of certain links in the categorial network. An example would be the utterance ‘*How many things are there?*’ where an item-based construction covering ‘*How many ?X are there?*’ and a lexical construction covering ‘*things*’ already exist, but where there is no link in the categorial network between the lexical construction for ‘*things*’ and the ‘?X’ slot in the item-based construction. In such cases, the

learning algorithm adds the missing link to the categorical network.

11.3.5 Entrenchment Scores

The constructions created by the learning operators have scores that reflect their entrenchment. During processing, higher scored constructions are preferred over lower scored ones. Upon creation, the score of a construction is set to 0.5. If used successfully, the score is increased by 0.1 and the score of other constructions of which the application would also have led to a solution is decreased by 0.3. The scores are bounded between 0 and 1. There is no built-in bias towards more general constructions. However, the fact that more general constructions are applicable in a broader range of situations and are therefore more frequently used, will, due to the dynamics of rewarding successful usage and punishing competitors, lead to higher entrenchment scores for more general constructions.

11.4 Experiments

This section presents a validation of our methodology for acquiring constructions on the CLEVR dataset discussed in Section 11.2. We first describe the experimental set-up (Section 11.4.1) and then present the evaluation results (Section 11.4.2).

11.4.1 Experimental Set-Up

The primary experiment consists in processing the 47,134 observations from our training set using the learning operators introduced above. For each experimental run, the observations are shuffled, so that any side-effects that might be caused by the order in which the observations are presented are levelled out. The learning operators are only active when an observation cannot be processed successfully by the constructions that have been learnt so far. Entrenchment scores are updated after each communicative interaction. The learning process is evaluated through four quantitative metrics: *communicative success*, *grammar size*, *number of constructions per type* and *active learning operators*. Communicative success is a binary measure computed by comparing the comprehended meaning with the gold standard annotation. In the graphs below, communicative success and active learning operators are plotted using a sliding window of 50 observations.

For completeness, we also present a secondary experiment in which the grammar learnt on the training set is evaluated on the test set. Communicative success is here averaged over the whole test set, and grammar size and number of constructions per type do not change during evaluation.

The experimental results reported below are based on 10 independent experimental runs. The error bars that are plotted represent percentiles 5 and 95.

11.4.2 Results

The results obtained through the primary experiment are shown in Figures 11.3 to 11.5. Figure 11.3 displays the communicative success and grammar size metrics respectively on the left and right y-axis as a function of the number of observations (x-axis). We can see that the communicative success starts at 0, as the experiment starts with an empty inventory of constructions. The degree of communicative success rises rapidly, with more than 90% of the observations being successfully processed by the learned grammar after only 500 observations have been

encountered. After 2000 observations, communicative success is already achieved in 99.6% of new observations.

The grammar size starts at 0 constructions and grows rapidly in the first phase of the experiment. After 500 observations, the grammar has reached its peak size of around 230 constructions that have some degree of entrenchment. This number then declines as a result of the rewarding and punishing of constructions. At the end of the learning process, the resulting grammar consists of 101.5 constructions on average.

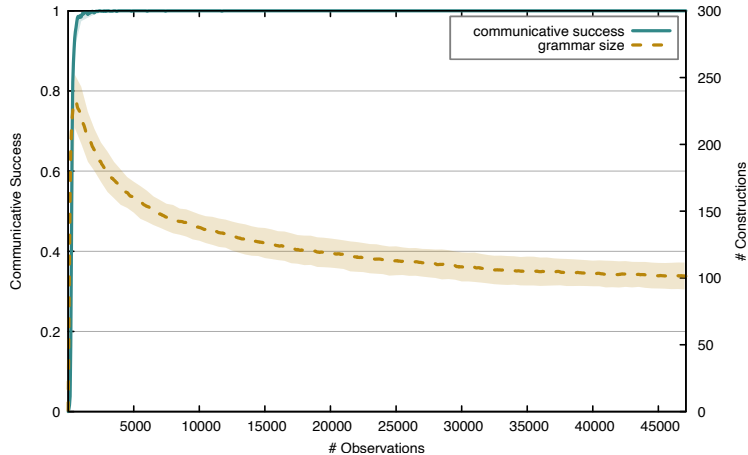


Figure 11.3: Evolution of communicative success (left y-axis) and grammar size (right y-axis) over time (full dataset).

An analysis of the types of constructions that are part of the learned construction inventory is provided in Figure 11.4. The results show that holophrase constructions flourish in the earliest phase of the experiment. In a second phase, item-based and lexical constructions take over the role of the holophrase constructions, with an abundance of item-based constructions being created. Over the course of the experiment, the linguistic inventory of the learner gradually reaches a stable state consisting of a limited number of entrenched lexical constructions and (more general) item-based constructions. At the end of the experiment, the grammar consists on average of 10.2 holophrase constructions, 57.1 item-based constructions and 34.2 lexical constructions. These results show that the holophrase constructions have not yet completely disappeared after 47,134 observations and that the theoretical maximum of 35 lexical constructions was attained in 7 out of 10 experimental runs. Note that it is the dynamic evolution of the number of constructions per type over time that is important, rather than the absolute number of constructions at a given moment in time.

Figure 11.5 shows the active learning operators over time, zooming in on the first 1000 observations. In the beginning, only new holophrase constructions can be created. Then, operators of the first class can generalise over these holophrase constructions and create new item-based and lexical constructions. After that, operators of the second class take over and create constructions based on partial analyses. In the final phase of the experiment, mainly operators of the third class, which only create new categorial links, are active.

We finally conduct a secondary experiment, which consists in processing all observations from the test set using the grammars resulting from the different experimental runs of the primary experiment. The average communicative success amounts to a perfect 100% in both the comprehension and production direction. The average grammar size amounts to 101.5 constructions, of which 10.2 are holophrase constructions, 57.1 are item-based constructions and 34.2 are lexical constructions.

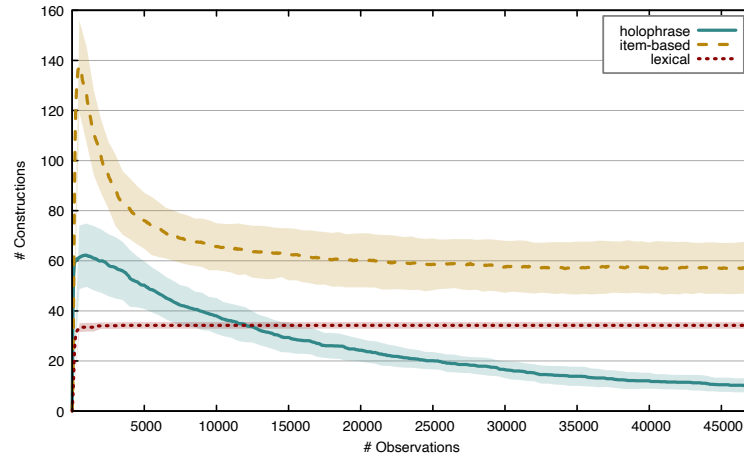


Figure 11.4: Evolution over time of the number of constructions per type with an entrenchment score > 0 (full dataset).

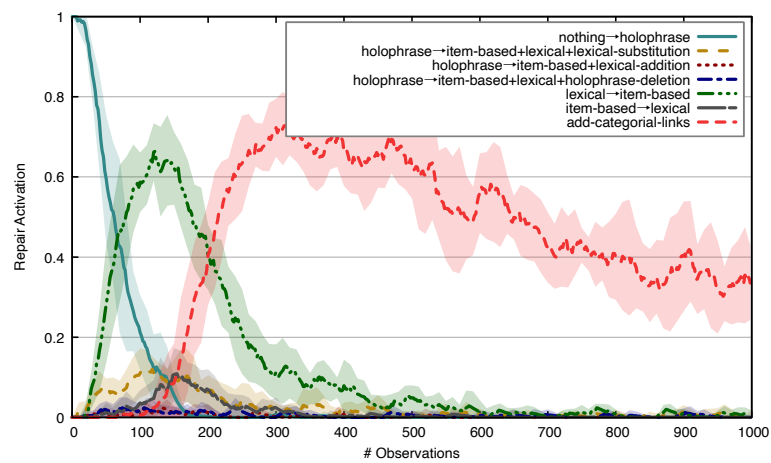


Figure 11.5: Active learning operators over time (first 1000 observations).

11.5 Related Work

Prior mechanistic models that operationalise the learning of constructions can be divided into two groups, based on the learning task that they address. A first class of models learns constructions paired with their meaning representation, either provided in the form of an annotated corpus [15, 7, 1] or obtained through task-oriented communicative interactions in a tutor-learner scenario [20, 4, 35]. A second class of models, as introduced by Gaspers et al. [18, 19], is designed to learn form-meaning pairings under referential uncertainty. As such, the exact meaning representations of the input utterances are not provided to the learning algorithm, but grammars are learnt based on the combination of input utterances and situational context snippets. In these experiments, input utterances always correspond to a single term in the situational context. In general, both classes of models have explored interesting ideas on a rather small scale, either because they were limited to specific linguistic phenomena [36, 20, 21, 4, 35, 33, 34, 41, 42, 40], or because of the limited morpho-syntactic and semantic complexity of the input utterances [12, 13, 14, 7, 18, 16, 17, 19, 1]. In all of the aforementioned work, either a segmentation of the input utterances, a lexicon or a set of predefined grammatical categories was provided. With the exception of the studies by Gaspers et al., the corpora that were used to learn and evaluate the models were not made available and were not described in sufficient detail to make reproduction and comparison feasible.

11.6 Discussion and Conclusion

The scientific contribution of the methodology and experiments presented here is twofold. On the one hand, they provide computational evidence for the cognitive plausibility of constructivist theories of language acquisition. These theories, as most prominently put forward by Tomasello [38], attribute the ability of children to acquire language to two main cognitive capacities: intention reading and pattern finding. Intention reading deals with reconstructing the intended meaning of observed utterances, while pattern finding implements generalisation processes that distil these reconstructed utterance-meaning pairs into abstract schemata embodying the linguistic knowledge of a language user. These schemata can then be used to fulfil the communicative function of language through the comprehension and production of natural language expressions. The methodology introduced in this work presents a mechanistic model of the pattern finding capacity. Based on utterances paired with a representation of their meaning, the learning algorithm gradually builds up an inventory of concrete to abstract form-meaning mappings, called constructions, along with a network of emergent grammatical categories that captures how the constructions of the grammar can combine to collaboratively comprehend and produce utterances. The experiments show that a small number of general learning operators, which become active if an utterance cannot be successfully processed by the grammar learnt so far, effectively leads to learning dynamics that are similar to those described in the psycholinguistic literature [31, 38, 2]. In the first phase of the learning process, the learner acquires holistic mappings between utterances and their meaning representation. Soon after that, holophrase constructions are generalised to item-based constructions that integrate a variable slot. At the same time, this generalisation process leads to the emergence of slot-filling constructions, here called lexical constructions. Along with the item-based and lexical constructions, a network of grammatical categories emerges, capturing the distribution of construction slots and their observed fillers. In a third phase, more abstract item-based constructions emerge, with an increasingly large number of variable slots. In the final phase of the learning process, most constructions have already been acquired and most remaining impasses can be solved by adding new links to the categorial network. The learning dynamics are influenced by the degree of entrenchment of constructions. Constructions that are often

successfully used become more entrenched, while their competitors are suppressed. As a result of this entrenchment process, the grammar reaches a stable state, while it remains adaptive to any changes in the discourse or environment. Similar dynamics have been observed in earlier experiments in the field of evolutionary linguistics, for instance in experiments on the emergence of compositionality in a population of autonomous agents [10, 43].

On the other hand, the methodology and experiments presented above pave the way for learning computationally tractable, large-scale, usage-based grammars that facilitate both language comprehension and production. The proposed learning algorithm supports online, interactive, incremental, transparent and data-efficient learning. The learner builds up its human-interpretable inventory of constructions and categories through the application of transparent syntactico-semantic generalisation processes. Already after a single observation, the fragment of linguistic knowledge acquired by the learner can be successfully used for language comprehension and production. As more and more utterance-meaning pairs are observed, the linguistic knowledge of the learner quickly expands and becomes better fit for achieving their communication goals. As a result of the dynamics of rewarding successful construction applications and punishing competing ones, the grammar of the learner remains ever-adaptive to any changes in the task or environment. Due to their online, interactive, incremental, transparent and data-efficient nature, we strongly believe that the proposed mechanisms for learning computational construction grammars can serve as an excellent basis for implementing the language acquisition ability of truly intelligent agents.

Chapter 11: References

- [DC1] Omri Abend, Tom Kwiatkowski, Nathaniel J. Smith, Sharon Goldwater, and Mark Steedman. Bootstrapping language acquisition. *Cognition*, 164:116–143, 2017.
- [DC2] Ben Ambridge and Elena Lieven. A constructivist account of child language acquisition. In Brian MacWhinney and William O’Grady, editors, *The Handbook of Language Emergence*, pages 478–510. John Wiley and Sons, Hoboken, NJ, USA, 2015.
- [DC3] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, 2013.
- [DC4] Katrien Beuls, Kateryna Gerasymova, and Remi van Trijp. Situated learning through the use of language games. In *Proceedings of the 19th Annual Machine Learning Conference of Belgium and The Netherlands (BeNeLearn)*, pages 1–6, 2010.
- [DC5] Katrien Beuls, Paul Van Eecke, and Vanja Sophie Cangalovic. A computational construction grammar approach to semantic frame extraction. *Linguistics Vanguard*, 7(1):20180015, 2021.
- [DC6] Jerome Bruner. *Learning to use language*. Oxford University Press, Oxford, United Kingdom, 1983.
- [DC7] Nancy Chang. *Constructing grammar: A computational model of the emergence of early constructions*. PhD thesis, University of California, Berkeley, Berkeley, CA, USA, 2008.
- [DC8] Alonzo Church. A set of postulates for the foundation of logic. *Annals of Mathematics*, 33(2):346–366, 1932.
- [DC9] William Croft. *Explaining language change: An evolutionary approach*. Pearson Education, Harlow, United Kingdom, 2000.
- [DC10] Joachim De Beule and Benjamin K. Bergen. On the emergence of compositionality. In Angelo Cangelosi, Andrew D. M. Smith, and Kenny Smith, editors, *The Evolution of Language. Proceedings of the 6th International Conference (EVO LANG6)*, pages 35–42, Singapore, Singapore, 2006. World Scientific.
- [DC11] Holger Diessel. *The acquisition of complex sentences*. Cambridge University Press, Cambridge, United Kingdom, 2004.
- [DC12] Peter Ford Dominey. Emergence of grammatical constructions. *Connection Science*, 17(3-4):289–306, 2005.
- [DC13] Peter Ford Dominey. From sensorimotor sequence to grammatical construction. *Adaptive Behavior*, 13(4):347–361, 2005.

- [DC14] Peter Ford Dominey. From holophrases to abstract grammatical constructions: Insights from simulation studies. In Eve Clark and Barbara Kelly, editors, *Constructions in acquisition*, pages 137–162. CSLI Publications, Stanford, CA, USA, 2006.
- [DC15] Peter Ford Dominey and Jean-David Boucher. Learning to talk about events from narrated video in a construction grammar framework. *Artificial Intelligence*, 167(1):31–61, 2005.
- [DC16] Judith Gaspers and Philipp Cimiano. A usage-based model for the online induction of constructions from phoneme sequences. In *Proceedings of the 2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 1–6. IEEE, 2012.
- [DC17] Judith Gaspers and Philipp Cimiano. A computational model for the item-based induction of construction networks. *Cognitive Science*, 38 3:439–88, 2014.
- [DC18] Judith Gaspers, Philipp Cimiano, Sascha S. Griffiths, and Britta Wrede. An unsupervised algorithm for the induction of constructions. In *Proceedings of the 2011 IEEE International Conference on Development and Learning (ICDL)*, volume 2, pages 1–6. IEEE, 2011.
- [DC19] Judith Gaspers, Philipp Cimiano, Katharina Rohlfing, and Britta Wrede. Constructing a language from scratch. *IEEE Transactions on Cognitive and Developmental Systems*, 9(2):183–196, 2016.
- [DC20] Kateryna Gerasymova and Michael Spranger. Acquisition of grammar in autonomous artificial systems. In M. Coelho, R. Studer, and M. Woolridge, editors, *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI-2010)*, pages 923–928, 2010.
- [DC21] Kateryna Gerasymova and Michael Spranger. An experiment in temporal language learning. In Luc Steels and Manfred Hild, editors, *Language Grounding in Robots*, pages 237–254. Springer, New York, NY, USA, 2012.
- [DC22] Adele E. Goldberg. *Constructions: A construction grammar approach to argument structure*. University of Chicago Press, Chicago, IL, USA, 1995.
- [DC23] Adele E. Goldberg. *Constructions at work*. Oxford University Press, Oxford, United Kingdom, 2006.
- [DC24] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. CLEVR. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2901–2910. IEEE Computer Society, 2017.
- [DC25] Philip N. Johnson-Laird. Procedural semantics. *Cognition*, 5(3):189–214, 1977.
- [DC26] Andrew Meltzoff. Understanding the intentions of others. *Developmental Psychology*, 31(5):838–850, 1995.
- [DC27] Richard Montague. English as a formal language. In R. H. Thomason, editor, *Formal Philosophy; Selected papers of Richard Montague*, pages 188–221. Yale University Press, New Haven, CT, USA, 1974.
- [DC28] Katherine Nelson. *Language in cognitive development: The emergence of the mediated mind*. Cambridge University Press, Cambridge, United Kingdom, 1998.

- [DC29] Jens Nevens, Jonas Doumen, Paul Van Eecke, and Katrien Beuls. Language acquisition through intention reading and pattern finding. In Nicoletta Calzolari and Chu-Ren Huang, editors, *Proceedings of the 29th International Conference on Computational Linguistics*, pages 15–25. International Committee on Computational Linguistics, 2022.
- [DC30] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank. *Computational linguistics*, 31(1):71–106, 2005.
- [DC31] Julian M. Pine and Elena V. M. Lieven. Slot and frame patterns and the development of the determiner category. *Applied Psycholinguistics*, 18(2):123–138, 1997.
- [DC32] Dan Sperber and Deirdre Wilson. *Relevance*. Harvard University Press, Cambridge, MA, USA, 1986.
- [DC33] Michael Spranger. Incremental grounded language learning in robot-robot interactions: Examples from spatial language. In *2015 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 196–201, 2015.
- [DC34] Michael Spranger. Usage-based grounded construction learning: A computational model. In *The 2017 AAAI Spring Symposium Series*, pages 245–250. AAAI Press, 2017.
- [DC35] Michael Spranger and Luc Steels. Co-acquisition of syntax and semantics: an investigation in spatial language. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 1909–1915, Palo Alto, CA, USA, 2015. AAAI Press.
- [DC36] Luc Steels. Constructivist development of grounded construction grammar. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 9–16, 2004.
- [DC37] Luc Steels, editor. *Design patterns in Fluid Construction Grammar*. John Benjamins, Amsterdam, Netherlands, 2011.
- [DC38] Michael Tomasello. *Constructing a Language: A Usage-Based Theory of Language Acquisition*. Harvard University Press, Harvard, MA, USA, 2003.
- [DC39] Michael Tomasello. The usage-based theory of language acquisition. In *The Cambridge handbook of child language*, pages 69–87. Cambridge University Press, Cambridge, United Kingdom, 2009.
- [DC40] Paul Van Eecke. *Generalisation and specialisation operators for computational construction grammar and their application in evolutionary linguistics Research*. PhD thesis, Vrije Universiteit Brussel, Brussels: VUB Press, 2018.
- [DC41] Paul Van Eecke and Katrien Beuls. Meta-layer problem solving for computational construction grammar. In *The 2017 AAAI Spring Symposium Series*, pages 258–265. AAAI Press, 2017.
- [DC42] Paul Van Eecke and Katrien Beuls. Exploring the creative potential of computational construction grammar. *Zeitschrift für Anglistik und Amerikanistik*, 66(3):341–355, 2018.
- [DC43] Remi van Trijp. *The evolution of case grammar*. Language Science Press, Berlin, Germany, 2016.
- [DC44] Remi van Trijp, Katrien Beuls, and Paul Van Eecke. The FCG Editor. *PLOS ONE*, 17(6):e0269708, 2022.

- [DC45] Lara Verheyen, Jérôme Botoko Ekila, Jens Nevens, Paul Van Eecke, and Katrien Beuls. Hybrid procedural semantics for visual dialogue. In *Workshop on semantic techniques for narrative-based understanding: Workshop at IJCAI-ECAI 2022*, pages 48–52, 2022.
- [DC46] Tom Willaert, Paul Van Eecke, Katrien Beuls, and Luc Steels. Building social media observatories for monitoring online opinion dynamics. *Social Media + Society*, 6(2), 2020.
- [DC47] Terry Winograd. Understanding natural language. *Cognitive psychology*, 3(1):1–191, 1972.

Part V

Interaction

Interacting with human-centric AI can encompass:

- an AI engineer examining tools to fix bugs or understand its output - which will be addressed in the section on Conceptual Shadows.
- an AI user trying to find out why a system is behaving in a certain way or the system trying to explain its behaviour to a user - which will be addressed in the section on Uncommon Ground.
- a scientist looking to find out how the system's properties and behaviour affect the users trust in the system - for which an approach is given in the section on Measuring Trust.

Chapter 12

Conceptual Shadows

Laura Spillner, Robin Nolte and Robert Porzel

Digital Media Lab, TZI, University of Bremen
Bremen, Germany

Abstract Word embeddings (high-dimensional vectors) are common input representations in NLP. However, this kind of representation is not meaningful to humans; it presents a black box that makes it difficult to explain how the vectors influence downstream models. Visualizing word vectors usually requires dimensionality reduction. We explore the visualization of word vectors as 2D images (one image per word, one pixel per vector dimension) by organizing the dimensions in the image with a self-organizing map. This method reveals new insights into how and where semantic information is encoded in the vector and allows us to pinpoint the source of downstream classification errors in the input representation. In this paper, we present the first results of an investigation into word embeddings that visualizes individual word vectors as images and explores what information the individual dimensions of the vectors encode. As this encoded information is specific to the given target concepts of a symbolic downstream classification task, it can be regarded as a projection from the symbolic space to that of the deep neural network.

Keywords: Word-Embeddings, Ontologies, Language Processing

12.1 Introduction

Undoubtedly, both symbolic and sub-symbolic approaches to artificial intelligence (AI) have their respective merits and individual shortcomings. In many applications, they are already joined at the hip, as the output of deep learning models often consists of classes that are symbolically described and used further on in some overall processing pipeline. One of the main areas of interest in the field of explainable artificial intelligence (XAI), and arguably one of the driving factors of recent interest in the field, is the explanation of *black-box* deep learning models. In natural language processing (NLP), deep neural networks (DNNs) are used in two ways: Firstly, to produce numeric input representations from natural language texts, and secondly, to solve downstream tasks, e.g., classification, clustering, or language generation. In many of these downstream tasks, conceptual models, such as ontologies, of the task-specific domain constitute the target representations for the classification.

For example, when classifying the part of speech (POS) of the words used in a sentence, specific classes are used as the values of the POS attribute, e.g., NOUN, VERB, or ADJ. These values are often part of a conceptual model, e.g., an ontology of linguistic entities, such as the GOLD ontology [7], the OntoWordNet model [10], or the LingInfo model [3]. In many cases, therefore, sub-symbolic approaches are used to classify entities stemming from some ontological model. In these sub-symbolic approaches, representations in which one word constitutes one symbol, such as bag-of-words or n-gram models, have largely been replaced by distributed semantic representations – also called *word embeddings* or *word vectors* – to represent text. It is generally accepted that the embeddings encode semantic information about a word and that words close to each other in the vector space are similar in meaning [17, 22]. However, high-dimensional word vectors pose difficulty from the XAI perspective because they essentially add a second black box, the model learning the embeddings, around the model used for the task itself.

When it comes to fields such as computer vision, many techniques have been developed to explain DNNs, e.g., by generating example images of the classes they are trained to identify or by highlighting image areas of particular importance in the classification [11]. Even though the input is represented numerically, the representation (the digital image) is still meaningful to humans. In contrast, a word vector as a point in very high-dimensional vector space is rather difficult to imagine or to represent visually. Because of this, visual explanations in the field of NLP usually fall into one of two categories: One option is to use dimensionality reduction to represent word vectors as points in 2D space, thus making it possible to see which words are close together. The other option is to consider not individual words but rather texts and highlight words as salient features, e.g., when predicting the topic of a text [4].

In this paper, we present the first results of an investigation into word embeddings that takes a different approach: We visualize individual word vectors as images and, inspired by XAI methods from computer vision, explore what information the individual dimensions of the vectors encode. This encoded information is specific to the given target concepts of the downstream classification task at hand. It can be regarded as a projection from the symbolic conceptual space to that of the DNN. For each conceptual entity, e.g., NOUN or VERB, CAT or DOG, etc., we obtain its visual projection into the sub-symbolic space. We call this the *conceptual shadow* of that entity. One application for this approach is to improve understanding of the input representations we use for NLP tasks. We hope to utilize this method to understand the origin of mistakes in the downstream model, such as incorrect classifications where a given ontological model constitutes the target representation. In the long run, this work seeks to connect sub-symbolic and symbolic representations of the same conceptual entity.

12.2 Related Work

In this section, we provide short overviews of prior art with respect to ontological models of linguistic knowledge, word embeddings, and explainable natural language processing.

12.2.1 Modeling Linguistic Knowledge

Various approaches have been proposed to model linguistic knowledge, i.e., the entities and features that make up human language, in formal ontologies. These approaches differ in some respects, such as alignment to upper layers, their modeling intent, and their scope. One point of divergence lies in the alignment to a foundational layer. While, for example, the GOLD ontology [7] aligns with the SUMO upper ontology [21], the OntoWordNet model [10] aligns with the DOLCE foundational ontology [16]. The LingInfo model [3] can be used with any foundational framework as it relies on meta-classes to model information about the lexical entities. For also

representing pragmatically relevant information, the SOMA-SAY [23] is based on Dolce Ultra Light and the Descriptions & Situations Module [9]. In contrast, the OntoWordNet aims at merging the linguistic information contained in WordNet with the respective classes employed in specific domain models, while both LingInfo and GOLD seek to incorporate more linguistic information, such as morphological and grammatical features of language. They all allow a direct connection of the respective linguistic information for terms with corresponding classes and properties in a domain ontology. Each model could be integrated into an NLP system as an additional module to allow reasoning about linguistic information or as a link between lexical and ontological resources.

12.2.2 Word Embeddings

Semantic embeddings have become standard input representations for many machine learning NLP tasks. Since the conception of word vectors in [17], improvements have been made with the introduction of character-based models and contextual representation [5, 1], which allow fine-tuning of pre-trained embeddings for downstream tasks [5, 24], as well as with the addition of transformer-based models [25] and attention mechanisms [28]. For this work, it is mainly important to differentiate between static representations, used in older models such as GloVe embeddings [22], and dynamic embeddings, which are part of Language Models like BERT [5]. With static embeddings, the same word is invariably represented by the same vector - it does not differ between different uses of the same word, e.g., homonyms or the exact spelling used as different POS. These static word vectors are then used as the input representation for downstream tasks. In contrast, when using dynamic embeddings, each use of the word in a text is represented by a different vector. Language models still represent each token in a text as a unique vector, but these are not generally intended to be accessible from outside the model.

12.2.3 Explainable NLP

The XAI literature differentiates between three types of explanations [4, 11]:

1. Explanations of network *processing*, including, e.g., Linear Proxy Models such as LIME [26]; salience mapping through occlusion [29]; etc.
2. Explanations of *representations* by probing the role of individual layers or individual neurons, for example, to generate images that maximize the activation of a given neuron and can be seen as prototypical examples of a given class [20, 19].
3. Systems that produce explanations.

Many works use explainable NLP in the third category to explain other models [6]. However, the focus of this work is different: Instead, we aim to explore on a deeper level where conceptual information is encoded in distributed semantic representations and which part of the information might be the cause for downstream symbolic predictions. Much of the work on explanations in NLP, especially when it comes to visual explanations, either utilizes dimensionality reduction or highlights salient features on the scale of words in a text [4]. However, it is not strictly necessary to reduce the dimensions of a word vector to visualize it. We tend to think of embeddings as vectors in high-dimensional space (e.g., 300 dimensions for GloVe embeddings) so that similar words are close to each other in this space. Yet a single word vector only consists of 300 numbers, while the numeric representation of an image might be made up of 6.000.000 numbers (a 1000px by 2000px RGB image). A word vector can easily be visualized as a kind of “barcode” of colors, with all 300 numbers arrayed in one dimension, the value of each number represented by the color. On this barcode, salient features (that is, the

most critical dimensions in the vector) can easily be highlighted. This method has been used previously to produce visual explanations for NLP tasks by [14].

12.2.4 Visualizing Word Embedding

The method presented in this paper is based on this same idea: Even though the individual dimensions of high-dimensional word embeddings do not obviously correspond to meaningful features to human eyes, they arguably still represent different features of what context a word usually appears in. By visualizing and analyzing these individual dimensions, we hypothesize that we can discover some clues as to *which* information is encoded *where* in the word embedding. A word vector can be visualized as a kind of “barcode” of colors - but to make it easier for the human eye to differentiate the individual dimensions, it might be helpful to visualize the same vector as an image, e.g., 300 numbers as a 300 pixel (15px by 20px) image. The main problem with this method is that humans will intuitively attribute meaning to the distance or closeness of individual pixels (e.g., “This area over there...”). This meaning, however, does not exist in reality, as the order of dimensions in the vector is random. Thus, we want to find a more meaningful organization of the dimensions of a word vector in a 2-dimensional space to visualize concept-specific areas of word embeddings as 2D images, so-called *shadows*.

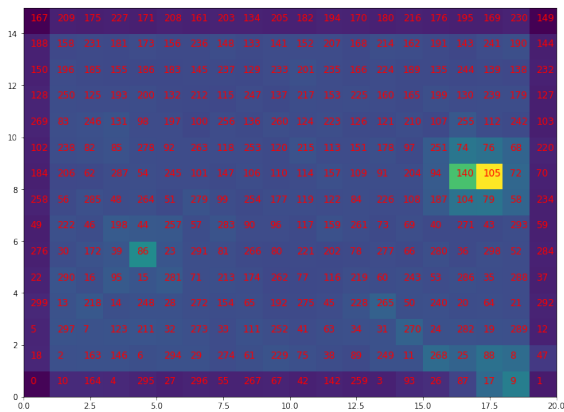
To organize where in the image the dimensions of the vector should be placed, that is, which pixel corresponds to which dimension, a self-organizing map (SOM) [13] presents an elegant solution. A SOM is trained on x examples, each represented by y features. The examples are then organized on a map. On these SOMs input vectors that are alike move closer together and ones that differ move away from each other by means of unsupervised clustering, i.e. learning vector quantization. When it comes to words represented by word embeddings, a naive approach would be to take for n words as examples and their k -dimensional word embeddings as their feature representation, which would result in a SOM that can place words on a map based on their embeddings (here, the SOM would be a method of dimensionality reduction). Our use case is different: We want to organize the dimensions in the embedding on a map. Thus, the k dimensions of the word vectors constitute the examples. The representations of these examples are the values of the dimension across the known words, meaning that each of the k examples will be represented as a n -dimensional feature vector.

By training a SOM with as many neurons as the word embeddings have dimensions, it is possible to arrive at a model in which each dimension is recognized by exactly one of the neurons. Using this SOM, a word embedding (e.g., of the word ‘the’) can be visualized as an image with as many pixels as there are dimensions in the embedding. Each pixel is colored based on the value of the dimension that is associated with the corresponding neuron on the map.

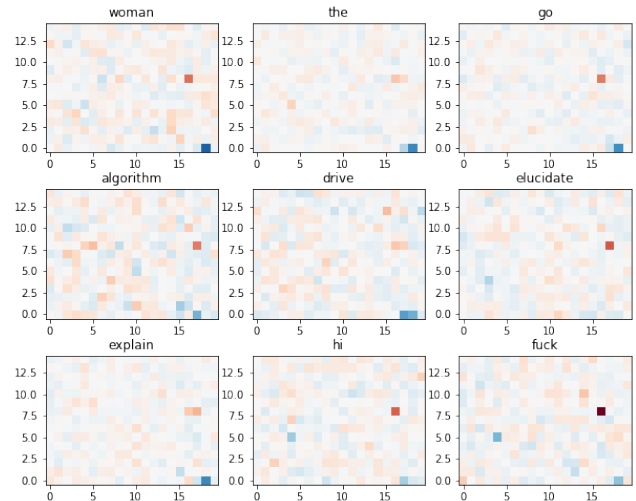
12.2.5 Projecting Static Embeddings

We first used this method to investigate static word embeddings: We analyzed the 300-dimensional GloVe embeddings provided by the open-source natural language library spaCy [12]. The SOM is used to arrange the 300 dimensions of the word vectors in a (small) 2D image. Thus the SOM provides a map encoding which pixel in the image represents which of the dimensions of the word vector. The pixel is colored based on the value of the word vector in the associated dimension. This means that the SOM is not used later for predictions on new examples - it is only used once to construct this dimension-to-pixel map, and is not required to generalize at all, as there are no other possible examples beyond the known vector dimensions.

We trained a 15x20 SOM on the 300-dimensional GloVe embeddings of 10.000 unique English words to analyze static embeddings. There are around 170.000 words in the English language,



(a) Visualization of the distance map of the SOM trained on static word embeddings. The map is comprised of 300 neurons, organized in a 15x20 map. Each neuron represents exactly one of the 300 dimensions of the embedding; overlaid in red are the numbers of the dimensions as they are ordered in the word vectors.



(b) A number of example word vectors visualized as images based on the SOM organization. Values around 0 are white, negative numbers red, and positive numbers blue. It appears that the most yellow dimensions identified in the SOM also have among the highest absolute values.

Figure 12.1: Visualization of static word vectors.

but not all have pre-trained GloVe embeddings. We constructed a dataset of words by first collecting all lemmas included in WordNet [18] through NLTK [2]. From this set, we identified the 64466 words included in spaCy as GloVe embeddings. Out of these, we took a random sample of 10.000 words on which to train the SOM, as we discovered through several trials that a corpus of 10.000 words is appropriate in terms of error and training time.

The training parameters of the SOM were adjusted empirically until the trained model arrived at a one-to-one matching of dimensions to neurons in the SOM (meaning that the SOM was able to correctly identify each dimension, as each neuron was trained to respond to exactly one of the dimensions). The trained SOM consistently achieved a quantization error of approx. 0.0005 over 2000 training iterations.

Figure 12.1 shows the layout of the trained SOM and a number of examples of words represented with the resulting layout. It stands to reason that those dimensions with values far from zero (positive or negative) contribute the most information, while those close to zero are less important. Therefore values at 0 are colored white, negative values are red and positive values are blue. The distance map of the SOM shows that there is overall very little variation, except for a few outliers located in three regions. These same regions can be found in the images showing a number of example words, where these pixels stand out in red or blue.

12.2.6 Analysis of Individual Words from Static Embeddings

Most interesting about the SOM shown in Figure 12.1a is that, while most of the neurons are relatively evenly spaced, there are several outliers - dimensions that are somehow more different from their neighbors than most. Most apparent are the pair of neurons corresponding to dimensions 140 and 105, the single neuron corresponding to dimension 86, and the cluster in the lower right corner. By comparing the map of the SOM in Figure 12.1a to the examples in Figure 12.1b, it becomes clear that the outlier neurons in the SOM correspond to those dimensions with greater absolute values than most.

word	140	105	r
‘coercive’	0.89	-3.15	3.15
‘plurality’	0.55	-3.12	3.12
‘minimise’	0.31	-3.11	3.11
‘deleterious’	0.48	-3.10	3.10
‘lessening’	0.20	-3.07	3.07
‘predetermined’	0.05	-3.05	3.05
‘concomitant’	0.90	-3.05	3.05
‘societal’	0.04	-3.03	3.03
‘constraining’	0.62	-3.03	3.03
‘quantifiable’	0.80	-3.04	3.02

(a) Highest r-values

word	140	105	r
‘fucking’	-4.21,	-0.04,	-4.17
‘ass’	-4.14,	-0.03,	-4.11
‘fuck’	-3.93,	0.08,	-3.93
‘wanna’	-3.94,	-0.02,	-3.91
‘song’	-3.90,	0.63,	-3.90
‘lol’	-3.87,	0.27,	-3.87
‘bitch’	-3.85,	0.02,	-3.85
‘cute’	-3.78,	0.38,	-3.78
‘ya’	-3.78,	0.29,	-3.78
‘movie’	-3.74,	0.40,	-3.74

(b) Lowest r-values

Table 12.1: Comparison of words at either end of the spectrum of r-values in the dataset. Duplicates (due to spelling differences and compounds like ‘ass-kisser’) are omitted. On the left (right) are the ten words in the dataset with the highest (lowest) r-value. Looking at this sample, it appears clear that those words where dimension 140 stands out are of an higher register, while the other group is decidedly informal.

Manual inspection of random words and their shadows (Figure 12.1b depicts a sample) revealed that in words of a comparatively high register (‘elucidate’), the right pixel (105) of the pair on the right stands out, and that in curse words, the left one (140) stands out strongly. We noticed that the two neurons 105 and 140, which appear as a pair in the SOM, never stand out together - it is always either one or the other that appears dark red. In some words (like ‘the’), neither stands out. Moreover, these two pixels often appear in dark red (negative value) but never in dark blue (positive value).

We inspected several synonyms of high-register words, such as ‘explain’ instead of ‘elucidate’, and found that neither pixel stood out for those. Furthermore, we also inspected several informal words such as ‘hi’, and found that in those, pixel 140 stood out almost as strongly as for curse words. We hypothesize that these dimensions capture the register of a word and act as opposites and calculated for each word x in the corpus an r so that:

$$r_x = \max(0, -x_{105}) - \max(0, -x_{140})$$

We then sorted all words by their r value. Those words with very low r-value are where pixel 140 is dark red while pixel 104 is neutral, and vice versa. Table 12.1 shows the ten words at either end of the list.

In the same way, we sorted the entire corpus of static embeddings based on the value at dimension 86 (a single pixel that stands out on the left of the map). This dimension apparently captures not the formality or register of words but instead seems to activate strongly if a word is likely to appear in a pornographic context. We tried the same with the cluster of pixels in the lower right of the map, both with individual pixels and combinations of the group. While there were some similarities, these were not as clear or meaningful as observed before (for example, sorting by 17 & 9 produced many words related to Catholicism on the one end, including, e.g., ‘antipope’, ‘tensured’, ‘archpriest’; and words which appeared related to customer service at the other, e.g., ‘management’, ‘service’, ‘customer’).

We also sorted the corpus of all words by their value in other random dimensions that do not stand out on the SOM, to assess whether these would also appear to indicate similar semantic explanations for their values. However, the lists of words produced from sorting by other dimensions had no seeming correlations or common characteristics.

We investigated the register-pair 140 & 105 further, testing what effect switching the value of

the two dimensions might have on a word. When taking, for example, a word of high register such as ‘elucidate’, switching the respective values of dimensions 140 and 105 results in a new 300-dimensional vector that does not belong to any known word. However, searching through the corpus of all words for the most similar word to this switched vector (in terms of cosine similarity of the vectors) results in the word ‘explain’. This connection holds for many words: applying the same technique to ‘sufficient’ results in ‘enough’, switching ‘corrosion’ leads to ‘rust’, ‘covertly’ to ‘secret’, ‘occur’ to ‘happen’, and so on - in a way, this can be used to find simpler synonyms.

12.2.7 Projecting Dynamic Embeddings

Analyzing the outlier dimensions in static word embeddings lead to some interesting insights. However, it seems that there are only very few dimensions that directly encode symbolic concepts such as register. For most other dimensions, the distance map of the SOM shows that there is very little variation, and their position in the resulting image is likely to be random. With dynamic embeddings, the word vector of a given word depends strongly on the context in which it appeared in the training text. Therefore, we examined whether visualizing these vectors might make it possible to investigate the results of word classification tasks such as POS tagging. If the same word can be used either as a verb or as a noun, somewhere in the vector, some information should be encoded as to which concept is more likely at hand in the given context. Our aim was that by visualizing dynamic word vectors with the SOM mapping, we might be able to find regions - that is, groups of dimensions - that are of particular importance for specific POS concepts.

SpaCy also provides a trainable part-of-speech (POS) tagging model, which consists of two layers: one takes a text and predicts dynamic, 96-dimensional embeddings for each token in the text, and the second predicts POS tags for these tokens based on the embeddings. We used these 96-dimensional word vectors to investigate dynamic embeddings. To train a SOM on static embeddings, we collected the pre-trained GloVe embeddings of a list of words. Due to the nature of dynamic embeddings, however, this is not possible here; an actual text is required since the conceptual representation of a word differs depending on its current context. Therefore, we used the Brown corpus [8] and generated the dynamic embeddings from spaCy’s pre-trained language model. As spaCy cannot process arbitrarily long texts, we only used the full sentences up to the 1.000.000th character. By removing punctuation and particle tokens, we obtained a dataset of 166.738 non-unique words with unique (dynamic) 96-dimensional word vectors.

First, we applied the same method as described above for static embeddings, training the SOM on the transposed matrix of word vectors. While there was some more variation in the SOM distance map, there did not appear to be any outliers as strong as in the static embedding map, and this method was not successful in differentiating between different POS concepts. Because of this, we took inspiration from two XAI approaches from computer vision research: the use of occlusion to analyze which features of the input representation are most important in the classification [29], and the generation of a prototypical image for a given class [19].

12.2.8 Masking the Shadows

By occluding parts of the word vectors, we hoped to find out which of the dimensions were actually necessary to recognize a word as a particular POS class, thus reducing the vector only to the essential areas. First, we tried this with words that the model had classified as a noun. For this, dimensions of the vector were occluded (set to zero) one by one, at each step

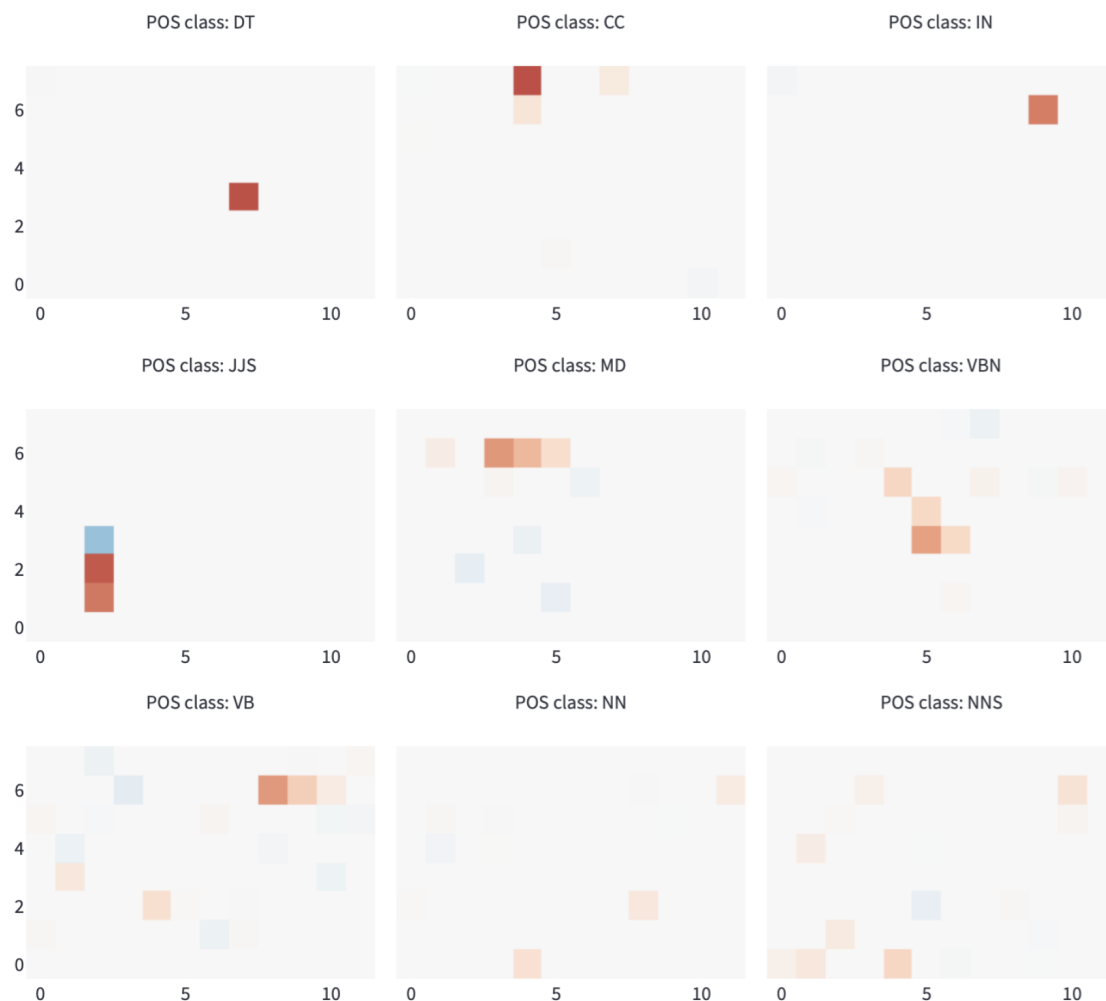


Figure 12.2: Average Shadows of POS Concepts

choosing the dimension of which the removal had the least negative impact on the probability of the vector being a noun. This was repeated until the probability dropped below 99% and then until it dropped below 50%. The first few removals increase the confidence in the noun classification instead of decreasing it. Testing this with a large number of words revealed that confidence in the noun classification usually stayed above 50% until only a few dimensions were left, sometimes as little as two. However, the remaining dimensions (visualized as pixels in the image) are not always the same, although many of the dimensions reappear over repeated tests.

We repeated the same process for all POS classes that the spaCy model identifies, which are based on the Penn Treebank classes [15]. The results are that most of the dimensions in a word vector are irrelevant for it to be classified as the same POS with above 50% confidence. This does not change if all but a few dimensions are reduced, when almost the entire vector is occluded, the prediction changes to a different class. Interestingly, NN (singular noun) appears to be the default classification: a vector with only 0s is classified as a noun, albeit with low confidence.

12.2.9 Projecting Prototypical Shadows

Next, we systematically tested the outcome of occlusion by reducing the vectors of all the tokens in the dynamic embedding corpus that the model had originally classified with high confidence, that is greater than 99%, until confidence dipped below 50%. For each POS class,

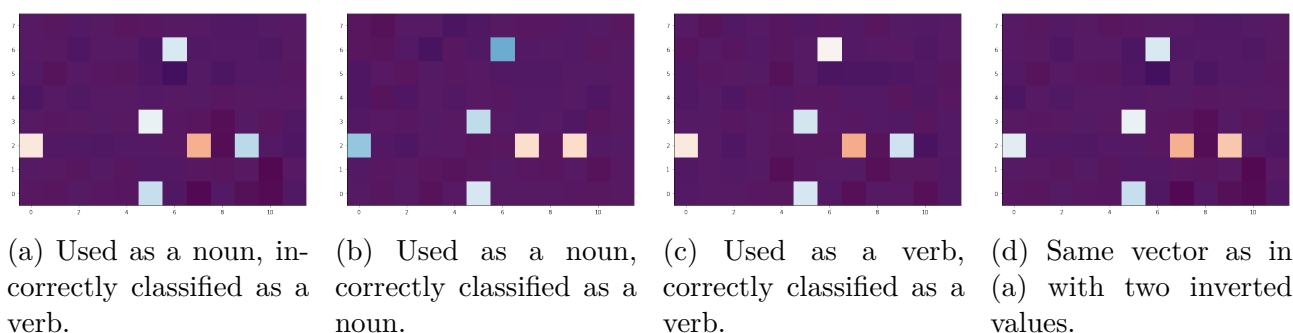


Figure 12.3: Four different vector representations of the word ‘garlic’, overlaid to highlight which pixels are especially relevant for classification as nouns. Here, we use a purple overlay over those dimensions which did not appear in at least 1/4 of minimal nouns.

we calculated the average of these reduced vectors. Figure 12.2 shows the resulting images for a selection of classes. In some cases, such as for DT (determiners), the result of the reduction almost always leaves the same dimension unoccluded, leading to an average image where only one or few dimensions appear very strongly. However, for others such as NN, there are many different possible results of the reduction. Thus, the average image is more translucent, and does not show a specific region. Classes such as MD, VB, or VBN (different verb forms) seem to be concentrated around different regions. It appears that for POS classes that can be considered conceptually more precise, there are only a few dimensions that are often or always very important for the classification. This is especially the case for those classes with a limited number of possible words, or which are marked by their form such as comparative or superlative adjectives. In contrast, concepts like nouns or verbs are more difficult to grasp.

12.2.10 Analysis of POS-tagging from Dynamic Embeddings

It appears that for POS classification, it is possible to identify areas in the vector images that are most important for the model to identify different POS classes. Therefore, we used these visualizations to investigate a problem that we had come up against repeatedly in prior work: models that are fine-tuned from pre-trained embeddings tend to struggle with very domain-specific language that differs from more standard texts. In particular, we have often struggled with the problem that recipe texts employ a kind of language that makes it difficult to identify the main verb of a sentence. This can be due to, for example, words being used as both verbs and nouns (e.g. ‘juice’), other words being left out (e.g. “chop tomatoes” instead of “chop the tomatoes”), missing punctuation, etc.

First, we decided to investigate one particular word, which we had stumbled upon in a previous study as an example that spaCy’s POS tagging model misclassified. We looked at three sentences containing the word ‘garlic’:

- (1) Add the garlic to the pan.
- (2) Add cauliflower and garlic mixture to the pot, mixing carefully to combine.
- (3) You have to garlic and salt the food.

In sentence (1), ‘garlic’ is correctly classified as a noun. In sentence (2), however, it is incorrectly classified as a verb with a probability of 49%, while the noun tag only has a probability of 46%. Sentence (3) is an example in which ‘garlic’ is used as a verb and correctly classified as such.

Figure 12.3 shows the visualizations of the three different vectors representing the word garlic in these three different sentences. As the confidence for the second version is already quite

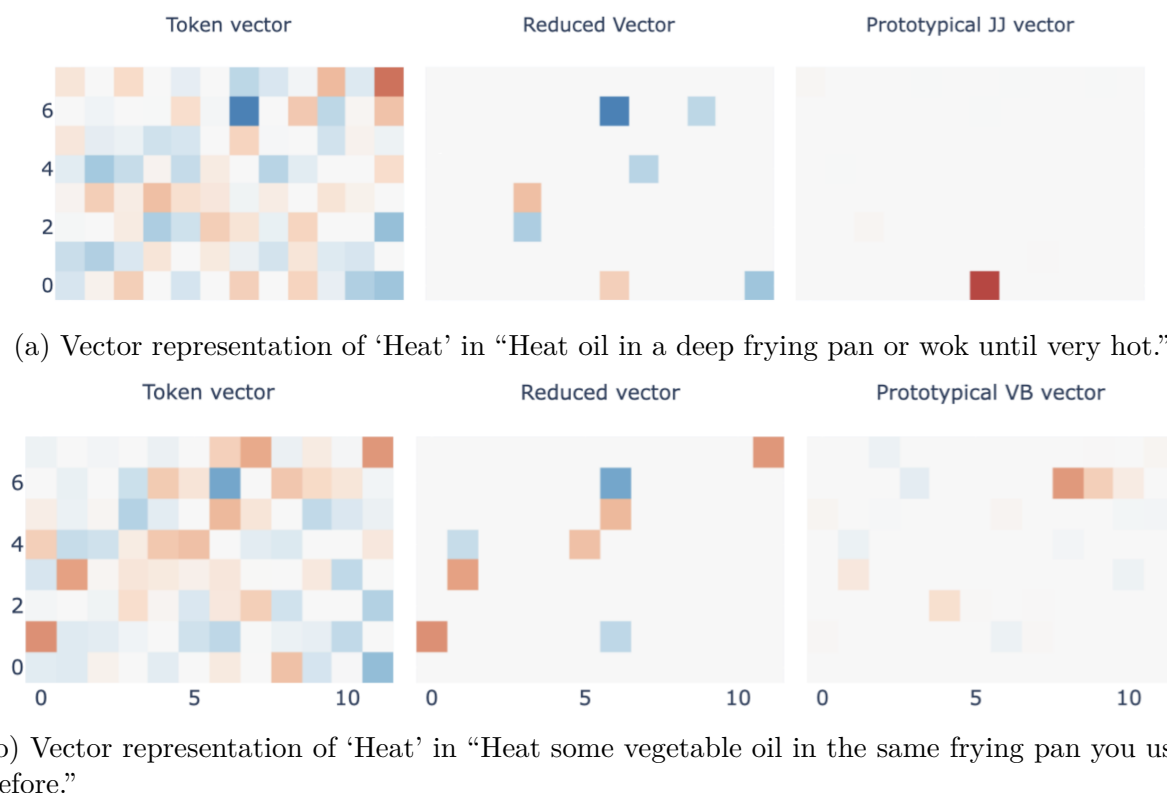


Figure 12.4: Two different vector representations of the word ‘Heat’. The three images show the entire embedding, then the same reduced to 50% confidence, and then the prototypical concept shadow of the POS class it was classified as.

low, and reducing the vectors would lead to different dimensions being left unoccluded, we did not reduce the vectors here. Instead, we masked most of the image, leaving those dimensions highlighted that were most often (across the whole corpus) unoccluded at 50% confidence.

It appears that the vector of the noun use of garlic, which was incorrectly classified as a verb (sentence (2), 12.3a), most strongly differs from the correctly classified noun (sentence (1), 12.3b) in the pixel on the far left at (0,2) and the one on the right at (9,2). Those two pixels are the same color in the vector representing garlic as a verb (sentence (3), 12.3c), opposite colors from the noun in 12.3b. Thus, we inverted these two pixels by multiplying their respective values with -1. Figure 12.3d depicts the result of these inversions. We used this ‘corrected’ vector as input for spaCy’s POS tagging model. As expected, the model now classifies this vector as a noun, with a confidence of 88%. This means that we were able to visually identify the exact dimension that was the reason for the incorrect classification of this token.

Next, we tried a slightly different approach with another problem, where a verb was incorrectly classified as an adjective, as seen in 12.4. As noun seems to be the default POS class, reducing the vectors of noun tokens leaves only very few dimensions unoccluded at 50%, and comparing them to the conceptual shadow shown in Figure 12.2 is not very helpful. However, this is not a problem for adjectives. We, therefore, considered two sentences:

(4) Heat oil in a deep frying pan or wok until very hot.

(5) Heat some vegetable oil in the same frying pan you used before.

The sentence (4), the first token ‘heat’ was incorrectly classified as an adjective instead of a verb. Therefore, we looked at the reduced vector of the token, as well as the conceptual shadows of the verb and adjective classes. Those were compared to a vector from sentence (5), where

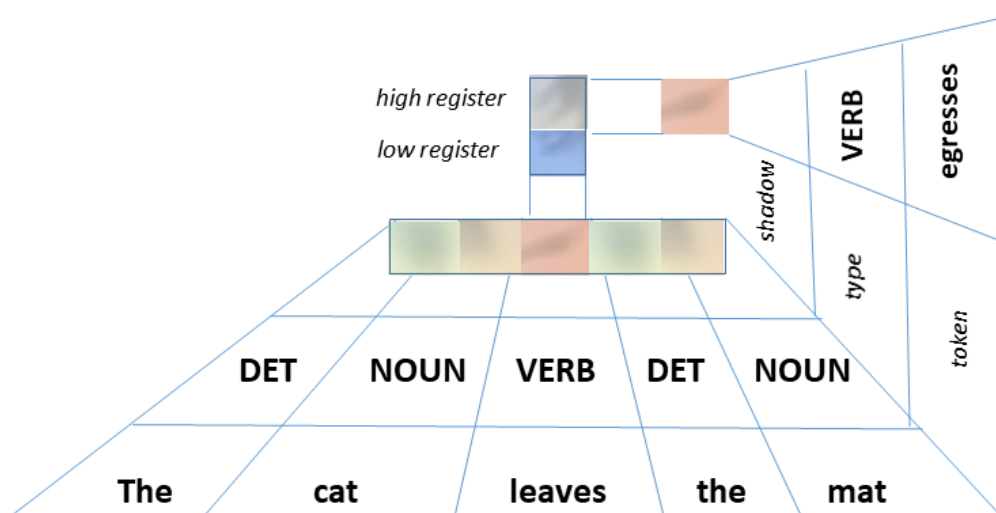


Figure 12.5: The general picture, showing a set of tokens in a given context, their conceptual types and the corresponding conceptual shadows (as illustrations).

the same word, ‘heat’, in the same position in the sentence, was classified as a verb correctly. Figure 12.4 shows these images. The two vectors that both represent the word ‘heat’ clearly share some features. Interestingly, many of the dimensions that are left in the reduced vectors are similar in both versions - clearly, small changes are enough to switch the classification from verb to adjective.

12.3 Discussion

The nature of this work is rather exploratory. The results of our experiments shed some light on how the meaning of linguistic concepts is encoded in high-dimensional word embeddings, which until now have been a black box in NLP that was quite securely closed. Addressing the cognitive elephant in the room, it is clear that human cognition is based on combinations of statistical processes together with increasingly symbolic generalizations over the extracted patterns. Some well-known phenomena such as prototypicality effects or radial categories [27] are out of the scope of most symbolic approaches, yet become quite easy to see in the conceptual shadows shown herein - where we can compare the shadows of very prototypical nouns and verbs to ones that are less *nouny* or *verby*. We are not aware of many other works which use visualizations of word vectors in their entirety, apart from the “bar code”-like images described in the beginning. By organizing the dimensions of these vectors on a map by training a SOM, we were able to identify areas of interest, as well as dimensions that appear to “belong together” as the pair of dimensions that seems to encode the register of a word. Together with the regions decoding POS, we can now form rudimentary ensembles of shadows that encode, for example, high-register nouns or vernacular verbs as depicted in Figure 12.5.

So far, this method has allowed us to identify small areas which appear to have recognizable tasks in the semantic representation, and to point out which of the dimensions of a word vector might be responsible e.g., for an incorrect classification. However, this in turn poses the question of why the dimension in question was “wrong” in the first place. To investigate this, we have to follow this lead one step deeper, and investigate what resulted in this particular weight when the vector was generated from the input text. One application for this work is to use it as a starting point from which to analyze downstream errors in NLP tasks and explain

their origins.

It is important to point out that any conclusion drawn from these visualizations is only ever related to the specific set of vectors on which the SOM was trained. A different kind of static embedding than GloVe might very well result in a very different map, with different outlier dimensions which might not appear to hold similar meaning to the ones we found here. This, however, is more feature than bug in our minds, as we visualize how a specific sub-symbolic system encodes conceptual dimension, which is – by its very nature – based on its training. In spite of the current limitations of the work presented above, we find that mapping the individual dimensions of word embeddings as a 2D image makes it possible to gather fascinating insights into the internal makeup of distributed semantic representations. We hope that this kind of low-level analysis of embeddings can serve as a starting point to gain deeper understanding of neural networks used in NLP and other symbolic classification tasks.

Chapter 12: References

- [EA1] Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In *Proc. of COLING 2018*, pages 1638–1649, 2018.
- [EA2] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”, 2009.
- [EA3] Philipp Cimiano, Paul Buitelaar, Anette Frank, Stefania Racioppa, Michael Sintek, Malte Kiesel, Massimo Romanelli, Berenike Loos, Thierry Declerck, Ralf Engel, Daniel Sonntag, Vanessa Micelli, and Robert Porzel. Linginfo: Design and applications of a model for the integration of linguistic information in ontologies. In *Proc. of OntoLex at LREC*, pages 28–32. ELRA, Mai 2006.
- [EA4] Marina Danilevsky, Shipi Dhanorkar, Yunyao Li, Lucian Popa, Kun Qian, and Anbang Xu. Explainability for natural language processing. In *Proc. of KDD 2021*, pages 4033–4034, 2021.
- [EA5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [EA6] Derek Doran, Sarah Schulz, and Tarek R Besold. What does explainable ai really mean? a new conceptualization of perspectives. *arXiv preprint arXiv:1710.00794*, 2017.
- [EA7] Scott Farrar and Terry Langendoen. A linguistic ontology for the semantic web. *GLOT International*, 7:97–100, 06 2004.
- [EA8] W. N. Francis and H. Kucera. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US, 1979.
- [EA9] Aldo Gangemi and Peter Mika. Understanding the semantic web through descriptions and situations. In *Proceedings of the ODBASE Conference*. Springer, 2003.
- [EA10] Aldo Gangemi, Roberto Navigli, and Paola Velardi. The ontowordnet project: Extension and axiomatization of conceptual relations in wordnet. In Robert Meersman, Zahir Tari, and Douglas C. Schmidt, editors, *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 820–838, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
- [EA11] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *Proc. of DSAA 2018*, pages 80–89. IEEE, 2018.
- [EA12] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

- [EA13] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.
- [EA14] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. In *2016 north american chapter of the association for computational linguistics*, pages 681–691. Association for Computational Linguistics, 2016.
- [EA15] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.
- [EA16] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. Wonderweb deliverable d18, ontology library (final). *ICT project*, 33052:31, 2003.
- [EA17] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.
- [EA18] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, nov 1995.
- [EA19] Anh Nguyen, Jason Yosinski, and Jeff Clune. Understanding neural networks via feature visualization: A survey. In *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 55–76. Springer, 2019.
- [EA20] Anh Mai Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. *CoRR*, abs/1605.09304, 2016.
- [EA21] Ian Niles and Adam Pease. Towards a standard upper ontology. In *Proc. of FOIS 2021*, page 2–9, New York, NY, USA, 2001. Association for Computing Machinery.
- [EA22] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- [EA23] Robert Porzel and Vanja Sophie Cangalovic. What say you: An ontological representation of imperative meaning for human-robot interaction. In *Proc. of JOWO 2020*, volume 2708 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2020.
- [EA24] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.
- [EA25] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [EA26] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ”why should i trust you?”: Explaining the predictions of any classifier. In *Proc. of KDD 2016*, page 1135–1144, New York, NY, USA, 2016. ACM.
- [EA27] Eleanor Rosch. Cognitive representations of semantic categories. *Journal of Experimental Psychology: General*, 104(3):192–233, 1975.
- [EA28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

- [EA29] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 818–833, Cham, 2014. Springer International Publishing.

Chapter 13

Understanding Uncommon Ground

Laura Spillner, Nima Zargham, Mihai Pomarlan, Robert Porzel and Rainer Malaka

Digital Media Lab, TZI, University of Bremen
Bremen, Germany

Abstract The need for explanations in AI has, by and large, been driven by the desire to increase the transparency of black-box machine learning models. However, such explanations, which focus on the internal mechanisms that lead to a specific output, are often unsuitable for non-experts. To facilitate a human-centered perspective on AI explanations, agents need to focus on individuals and their preferences as well as the context in which the explanations are given. This paper proposes a personalized approach to explanation, where the agent tailors the information provided to the user based on what is most likely pertinent to them. We propose a model of the agent’s worldview that also serves as a personal and dynamic memory of its previous interactions with the same user, based on which the artificial agent can estimate what part of its knowledge is most likely new information to the user.

Keywords: Deep Learning, Explanations, Language Processing

13.1 Introduction

One major motivation of explainable artificial intelligence (XAI) is the desire to make the predictions of black-box machine learning (ML) models more transparent [4]. Adadi and Berrada conducted a survey of the XAI literature and created an overview of common XAI methods - all of the methods presented range over the internal mechanisms of a ML model [2], which we will refer to as *introspective* explanations.

More recently, there has been an increasing need to explain AI behavior to non-expert users of AI systems. Therefore it has been proposed that XAI needs to focus more on end-users as the recipients of the explanations [24, 14] and on how XAI methods can be evaluated from a human-centered point of view [19]. This sheds a different light on estimating the quality of an explanation and how it can be tailored adequately to the given user. This view is supported by the argument made by several researchers in the field that explanation should be understood as interactive conversations [17, 8].

The usefulness of explanations can depend on the context in which they are given, including the situation and the user, as pointed out in several recent studies [7, 10, 19]. To foster a

human-centered approach to XAI, individuals and their preferences should be considered by appropriate user models. These can encompass, e.g., a level of education, cultural background, familiarity and interest in technology, perception of the agent, mood, and social settings [26]. To accommodate these factors, personalization of AI explanations might be crucial [7, 10]. This is particularly relevant in domains where users interact with the same artificial agent over a long period, such as smart home devices, robots in household and care domains, and other personal AI applications. If users ask for explanations because they expected different AI behavior, there must be some disagreement between users and AI in their respective beliefs and knowledge, which is what we call their *uncommon ground*. Explanations should allow the user to realize where the uncommon ground is. In error cases, this can empower users to correct the AI’s behavior for future interactions; in other situations, it can enable them to see that they are missing important knowledge themselves. To facilitate this, explanations by the AI should reflect its experience from previous interactions with the same user. Personalization can improve user satisfaction [23, 25] by using user models created with direct or indirect user input, i.e., user entry or automated systems that adapt to user behavior [18]. Although direct user inputs can have advantages, an agent that is not properly configured may be less acceptable [5], thus personalization through indirect adaptation might be preferential [28].

We argue that it is thus necessary to consider how to generate *extrospective* explanations: explanations that are given with respect not only to the system itself but also take into account what the system knows about the user’s expectations and draw on experience from earlier interactions. This extrospective perspective on XAI is at the heart of our work, as we focus on domains where users interact with smart software, intelligent devices, and autonomous agents/robots repeatedly over time, e.g., smart home device [citation redacted] and household robotics [citation redacted]. This, in a sense, entails that users have their ‘own’ personalized AI. As Alizadeh et al. point out, explanations in the context of repeated interactions have not been studied as much [3]. Therefore, we seek to examine this challenging problem systematically.

13.2 Explanations in Context

To evaluate explanations, it makes sense first to consider *why* there is a need for an explanation from the user’s side. We argue that in domains such as household robotics, where non-expert users commonly interact with AI systems that they generally trust, if a user asks for an explanation in a specific situation, this will likely be because the AI’s behavior or recommendation differed from the user’s expectation - meaning the need for explanation is motivated by the user’s *surprise*. The XAI literature commonly differentiates between global and local explanations: Global explanations aim to make the overall workings of a (black-box) AI model transparent [2, 4], e.g., through global feature importance or approximation. Here the question is about “how” the model works [14]. Local explanations try to explain a specific instance of model prediction or agent behavior [2, 4], e.g., through local feature importance or salience. Here the question is about “why” the model arrived at this prediction [14]. Global explanations are independent of any particular prediction. Regarding local explanations, we can further categorize explanation requests based on whether or not the user expected a different AI prediction. When users ask for a local explanation for one specific prediction, often the reason is that they expected a different outcome in this instance.

Liao et al. surveyed AI practitioners and developed a taxonomy of questions that users ask of XAI systems [14]. They identified several situations and goals, ranging from users wanting to increase their confidence in their own decision to trying to debug or improve the AI system. Both global and local explanations can be useful in cases where one wants to better understand how a model works, what its prediction is based on, or make its workings transparent for

ethical purposes. This is often the goal of researchers or developers but was also a common need by other practitioners surveyed by Liao et al. However, as the authors also point out, for non-expert users of AI, the need for local explanations most often arises out of surprise [14].

There are many domains in which AI is supposed to support humans in making decisions or achieving tasks, such as AI software to support medical diagnosis [11], but humans have to make the final choice. Even though in many such domains AI performance equals or even surpasses human performance [11], it is still often desirable for the human expert to make the final decision on whether or not to trust the AI prediction. If the AI prediction does not align with what the user expected, it is crucial that the AI is able to explain itself [27, 14]. In this case, the user’s goal is to understand whether their intuition or the prediction of the system is more likely to be correct. In domains where users can easily recognize AI errors, e.g., everyday activity domains such as household robotics [12], they similarly ask for explanations in (presumed) error cases to find the source of the error. Here, the user’s goal can be to adapt their behavior to the system’s weaknesses or improve the AI’s performance, e.g., by giving it additional information.

Suppose users ask for explanations because the AI’s prediction or behavior violated their expectations. In that case, a good explanation clarifies why the AI came to a different conclusion than the user. The explanation given should help users in finding the source of the disconnect. In some domains, this will enable users to understand AI errors or correct agent behavior in the future. In other domains, understanding why the AI prediction did not align with their intuition might help humans to realize if and why their intuition was wrong and decide whether or not to trust the AI prediction instead. The use case we describe essentially calls for *counterfactual* explanations: instead of simply asking “why?”, the user asks, “why not [what I expected]?”. This is supported by Miller et al.’s argument that when humans ask for explanations, there is often an implied counterfactual that is not stated explicitly [17].

Consider a situation in the household robotics domain where a robot has helped its human user to bake a cake for a birthday party. The robot knows that because the cake has a lot of icing, it must be stored cool until the party later in the evening. The robot finds that the cake is too big to fit in the fridge, where things that must be kept cool are usually stored, but as it is cool outside, it puts the cake (in its container) on the patio table. If the human asks the robots “Why did you put the cake outside?”, what are they really asking? The literal answer to the question is “the cake needs to be kept cool”. However, the human is aware of this - what they are really asking is likely “Why did you put the cake outside, instead of where I expected it [e.g., in the fridge]?” This is the counterfactual assumption that is asked for - the user wants to know how the situation would have to be different for the cake to be stored in the expected location (e.g., if the cake were smaller, it would fit in the fridge).

There have been a number of works recently that aim to generate counterfactual explanations in different use cases [17, 15, 22, 13]. Yet, these approaches often encounter a problem when a) there is more than one possible counterfactual and b) the counterfactual case, that is, the prediction or behavior expected by the user, is only implied and unknown to the AI. If the decision to be explained was a choice between two options, then a counterfactual explanation simply needs to state what would have caused the decision to flip from what was actually chosen to the alternative. However, if there are more than two options such as A, B, and C, then generating a counterfactual explanation involves choosing which alternative case should be looked at. If the user explicitly states which alternative they expected, this can be used as the basis for the counterfactual explanation. However, in many situations, humans will not provide this information explicitly but only imply it or consider it obvious (it is often obvious to humans due to context knowledge, but not to the artificial agent). We can consider several

strategies to discover the counterfactual: the simplest approach is to explicitly ask users what other outcome they expected. Alternatively, it is possible to try to predict the counterfactual case, e.g., by constructing a mental model of the user’s knowledge or by collecting training data from a variety of different situations. In this paper, we propose to tailor the explanation in order to present to the user that piece of information that is most likely to be surprising to them.

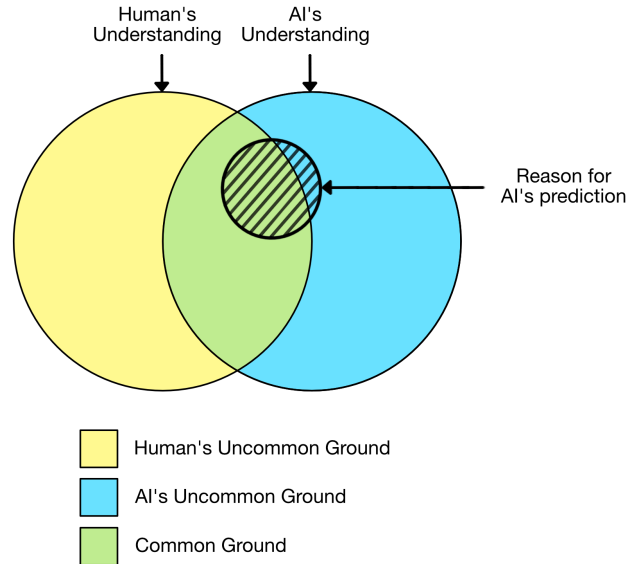


Figure 13.1: The left circle represents the human’s understanding, that is, their knowledge and beliefs, worldview, values, etc., while the right circle represents that of the AI. Their intersection (green) is the common ground, i.e., knowledge both parties share. That part of their knowledge and understanding that is not shared by the other agent is the human’s and AI’s respective uncommon ground. The lined circle represents the subset of the AI’s knowledge used in its reasoning to arrive at the current prediction. In the situation shown here, most of the knowledge used for the prediction is shared by the human (the part of the lined circle overlaying the green area), but the AI also reasoned with some facts or beliefs which the human does not agree with (the part of the lined circle that is in the blue). Because of this, the AI arrives at a different prediction than what the human expected.

13.3 Uncommon Ground

Both the human agent and the AI or artificial agent have some understanding of the world and the current situation. In the following sections, we will consider the case in which this understanding is modeled explicitly and AI decisions are based on symbolic inference and reasoning. Although the framework we present could be applicable to ML models as well, our contribution is based on the symbolic approach. In the household robotics domain, we usually model the AI knowledge in an Abox (“assertions”: statements about concrete items in the current situation) and Tbox (“terminology”: concept definitions, general knowledge). In the following sections and the given example, we will use a simplified model in which the knowledge about the world, in general, contains rules about how the world works (e.g., “if an item is a cake, then its taste is likely sweet”), while the knowledge about the specific situation contains facts about it (e.g., “item A is a cake”). Therefore, we will assume that the agent’s and the

human’s understanding of the world can be modeled as such facts and rules, and that both of them are able to arrive at new conclusions about the situation through logical reasoning.

Presumably, the respective world views of humans and AI mostly overlap. Yet, when the need for explanation arises from surprise on the user’s side, there must be at least some small area where they do not agree: we will call this their *uncommon ground*, as opposed to the common ground that is their shared knowledge. In language use, the importance of *common ground*¹ has been well established [6]. In the case of AI systems using logical reasoning, when there is disagreement, this means that somewhere in the chain of reasoning of the agent, there must be a fact or a logical step with which the human would disagree (Figure 13.1). Note that for the purposes of this framework, we consider beliefs, values, moral guidelines, cultural standards, etc., to be part of this knowledge and count them as ‘facts’ or ‘rules’ from the the agent’s point of view, even if they do not constitute (subject-independent) facts about the physical world. Thus, the uncommon ground that leads to unexpected behavior could be that one agent is not aware of some property of the current situation that the other knows, such as “we are out of milk”, but it could also be a difference in what they consider polite, good, or proper, such as that one of them does not agree with the ‘rule’ that “guests should remove their shoes before entering a home”. Even though the agent does not know what outcome or prediction the user expected, the part of their decision-making that belongs to the uncommon ground must necessarily be the explanation for the counterfactual: if *this* had been different and had aligned with the belief of the user, then the prediction of the agent would have been the same as the prediction of the user.

Consider again the example of the birthday cake introduced in the previous section. If the human knows that the cake should be put in a cool place, but they expected the robot to put it in the fridge, then they were unaware that the cake is too big to fit into the fridge. This is the uncommon ground in this scenario: the robot knows a rule (items of this size do not fit in the fridge) that is not part of the human’s knowledge. In this case, this is not because the human disagrees with this rule but simply because they are unaware of it. We can think of other examples where instead, the human disagrees with something the robot believes: maybe they are aware that the cake is too big for the fridge, but they don’t consider the outside an appropriate place to store a cake. In this case, they would likely want to correct the robot’s behavior for the future. Finally, a third situation would be one where the human is unaware of a certain property of the current situation: perhaps they do not know that outside on the patio, it is quite a bit cooler than inside because the previous days were much warmer. The robot knows a fact (the temperature on the patio) that the human has not perceived yet. This poses the question: How can the robot guess which situation is more likely, and where the uncommon ground is?

We propose a model of the agent’s worldview that also serves as a personal and dynamic memory of its previous interactions with the same user. Based on this memory, the agent can judge what part of its knowledge is more likely to be shared by the user. We base our model on the SUDO model proposed by Porzel, in which there are four areas of context (Figure 13.2) [20]:

- **Situational Context:** Includes knowledge specific to the current situation, such as the facts observed by the agent. In knowledge engineering, this is termed the Abox and

¹It is important to note that *common ground* is not the same as *common knowledge*. While the knowledge in the common ground is shared between both human and AI, they do not necessarily know that the other shares this knowledge. In epistemic logic, common knowledge goes beyond common ground, describing knowledge that is not only known by all in the group but also known by all to be known by all. Thus, the common knowledge of the two agents is a subset of their common ground.

represents instances.

- **User Context:** Includes the knowledge about the user, their physical or cognitive abilities (e.g., young children or elderly users), current activity, physical or emotional state (e.g., stress, sadness, fatigue), their personality or preferences, (e.g., on privacy, agency, etc.) as well as other humans (e.g., family, guests).
- **Discourse Context:** includes information on previous interactions with the user: their requests, the AI's predictions or actions, and the user's behavior and reactions.
- **Ontological Context:** In contrast to the situational context, this includes the general understanding of the world, such as commonsense knowledge or rules that can be used to reason with; also known as Tbox.

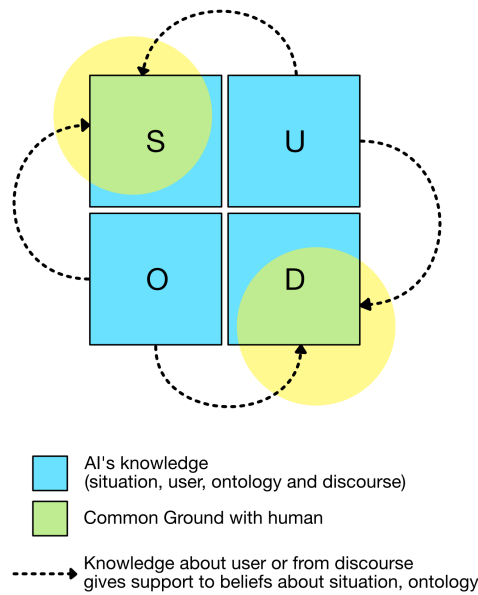


Figure 13.2: The internal model of the agent's knowledge is divided into **situational context** (S), **user context** (U), **discourse context** (D), and **ontological context** (O). The knowledge and beliefs of the user also include a situational context and an ontological context (yellow), which overlap with those of the agent, but are unknown to the agent. Their respective intersections represent what the agent and the user agree on. Somewhere in the set difference (the uncommon ground) must be the reason for the user's surprise. Information from both user context and discourse context can give support (dotted arrows) to facts stored in the situational and ontological contexts, allowing the agent to reason that these facts are less likely to be part of the uncommon ground.

The reason for the uncommon ground might be that either agent or user is missing some knowledge, or have specific beliefs that the other does not share (which may or may not be wrong, or might be a matter of perspective such as social etiquette). Many works in XAI focus on explanations in the context of errors of the AI, e.g., Alizadeh and Pins developed a taxonomy of different situations in which users ask for an explanation based on why an intelligent assistant made a mistake [3]. However, if the disconnect between agent and user is based on the uncommon ground in the situational context, this does not necessitate an error by the AI: either agent or user might be basing their decision on outdated knowledge; or there might be a property of the situation at hand that is not available to or perceivable by the AI. Similarly, a disagreement in the ontological context does not necessarily need to be a bug, but

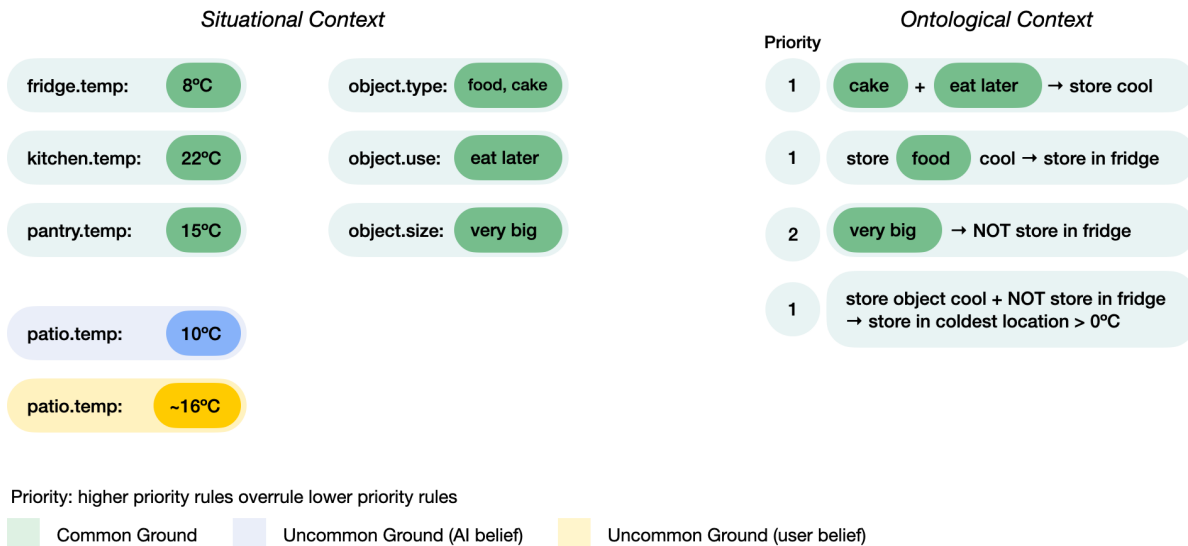


Figure 13.3: A simple representation of what (a subset of) the situational and ontological contexts of the AI’s and the human’s knowledge in the birthday cake example (see section 2) could look like. The situational context contains facts that the agents believe about the world’s current state, including the properties of objects or locations. The ontological context includes rules describing how the agents believe that the world works and how they should act in certain situations; e.g., when a cake has to be saved for later, it should be stored in a cool location and not left out in a warm room. In this model, rules can contradict and outrank each other. A rule with priority level 2 precedes one with priority level 1. In this case, the rule that very big items cannot be stored in the fridge outranks the rule that food items should be put in the fridge to store cool. Here, the only difference in their respective worldview is that the AI agent is aware that it is quite cool outside today.

instead might be because of a cultural preference by the user (e.g. “guests should keep their shoes on in my house”) that does not align with the default programming of the AI, and which is not shared by many other users.

In the ontological context, we considered three layers: firstly, the AI will have some fixed knowledge that cannot be edited by a user, e.g., basic rules to prevent unsafe agent behavior. Other rules, forming the second layer, might be open to editing: social customs or common wisdom might be “pre-installed” on a household robot, but if it conflicts with the user’s beliefs, could be changed based on their preferences. An example of this is a household robot by a western company that believes that meals are eaten sitting on chairs around the table, but this rule could be changed by users from other cultural backgrounds who prefer to eat sitting around a table on the floor. On the third layer, users have the option to teach an intelligent device completely new knowledge, as long as it does not conflict with any previous beliefs but was simply not available to the agent at all.

Figure 13.3 illustrates what the respective sets of knowledge of both a human and an AI agent in the birthday cake example might look like. Of course, this is only a small subset of what either agent knows about the world and the situation at hand, we focused here on those facts and rules which are directly relevant to the situation and did not consider different levels. In our example, the agent knows or rather believes a number of facts about the object at hand (the birthday cake) as well as the possible storage locations (fridge, different rooms, patio). The rules can be understood to illustrate the reasoning process of the AI in this situation when read from top to bottom. Let us consider a situation in which the human agrees with almost

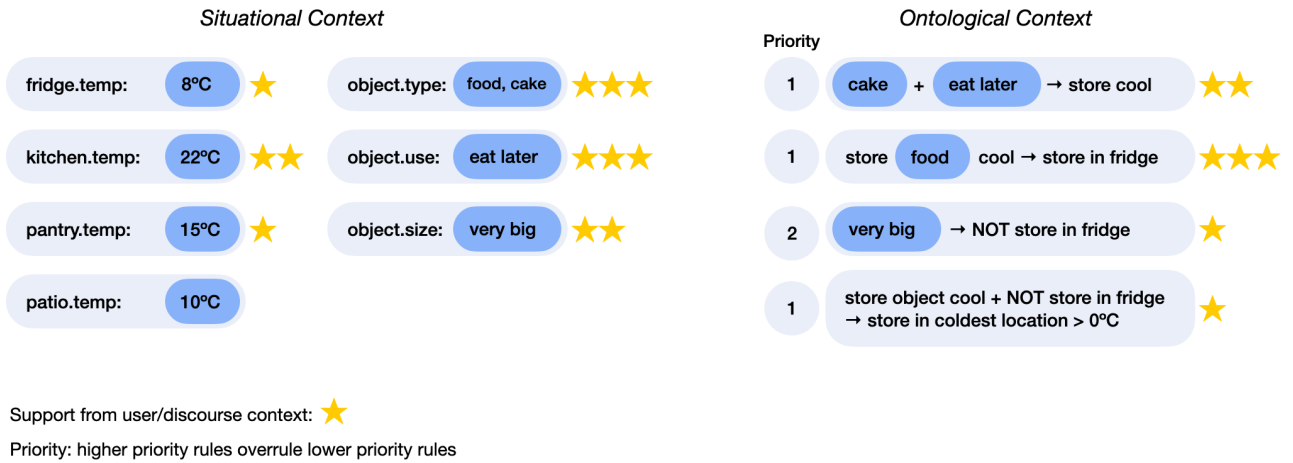


Figure 13.4: The same example situation as shown in 13.3, but from the point of view of the AI agent. The AI does not know what the human believes or where the uncommon ground is. The stars indicate that a piece of knowledge has received support from what the agent knows about the user or from previous interactions with the user. In this case, three stars in the situational context indicate that a fact has been communicated to the robot by the user, two stars indicate that the robot knows that the user has perceived this fact as well, and one star indicates that this is a common fact which the user has agreed with in past interactions. In the ontological context, three stars would be a rule that the user had explicitly supported, while two stars indicate a rule that the agent knows the user has employed themselves in the past, and one star indicates that the agent had successfully used the rule in the past when the user did not disagree with its actions.

all of the rules and facts that the AI believes, but is not currently aware of the fact that the outside temperature is quite cool. If they were made aware, they would come to the conclusion that the robot is in fact right to put the cake on the patio. Instead, they had expected it to place the object in the hallway, which they thought was just as cool as the outside. In this case, the uncommon ground was due to incorrect knowledge on the side of the user, and not due to AI error.

In our framework, the situational and ontological contexts are influenced by two further contexts that describe how the agent understands their relation to the user as another intelligent agent. The user context and discourse context impact the knowledge in the situational and ontological context: When the user corrects the agent’s beliefs, makes it aware of new facts, or adds their own preferred rules, the knowledge that the agent has is changed based on their interaction. Facts asserted and rules edited by a user become common knowledge between the agent and user, meaning that the agent knows that the user shares this knowledge and can, in the future, assume that this is not part of the uncommon ground. Moreover, the information from the user context and the discourse context can provide *support* for the knowledge in the ontological or situational context: If some facts or rules are used in successful interactions, then the knowledge that the AI used to arrive at this decision is likely knowledge that is shared by the user. Therefore, we propose that the agent’s internal model should be able to represent this kind of context-based support of existing knowledge (see Figure 13.2), e.g., by adding weight to existing connections. In Figure 13.4, we illustrate the same knowledge space as in Figure 13.3, but from the point of view of the AI. The AI agent does not know which part of its knowledge is shared with the user, where the uncommon ground is, or what the human believes. However, their beliefs have gained support from previous interactions with the user as well as their knowledge of the user’s previous actions and statements. The support is illustrated

by stars on the respective rules and facts. To keep the graphic simple, we do not distinguish between user and discourse context. For this example, we imagine that all rules and facts have received some support except the one that describes the temperature outside (as the user is wrong about this fact, it cannot have received any support by interactions with the user). The other facts and rules have received varying levels of support, depending on whether or not the information was explicitly communicated by the human (e.g., the robot knows for a fact that the human knows that the item is a cake that is to be eaten later), is based on observations (e.g., the human has been in the kitchen. Therefore the robot can assume that the human knows the temperature there), or is based on previous successful interactions (e.g., the robot had previously put food items in a different location when they did not fit in the fridge, in that case, there was no problem).

13.4 Generating Extrospective Explanations

Using the approach of modeling shared beliefs, we differentiate between two kinds of explanation-generating processes: Typically, when AI models or intelligent agents are asked to explain their predictions or behavior, generating this explanation is an introspective process. By that, we understand that the explanation is based on the AI’s internal model, set of beliefs, or knowledge base. However, if an artificial agent also considers what it knows about the user’s (another intelligent agent) beliefs, this becomes an *extrospective* process.

When generating local explanations for end-users, we can assume that it is usually not possible to present an explanation of everything that influenced the decision or the entire chain of reasoning of the AI. Human-centered explanations ought to be short and relevant and not include superfluous or obvious information [17, 16, 8, 14]. When choosing what information is relevant, the standard (introspective) approach is to choose those aspects that have the most significant influence on the prediction, e.g., the most salient part of the input features [1, 2, 21, 9]. When working with agents that reason logically, one might choose the last logical step in the reasoning process or the most specific rules².

In the birthday cake example, there are several introspective ways to choose which piece of knowledge or belief should be presented to the user if they ask why the robot put the cake on the patio. One reasonable option is to present the fact about the situation that had the most significant influence on the decision, e.g., by looking at which statements were used most often or the last step in the reasoning process. We can present the last step in the robot’s reasoning process since that is closest to the final decision: “as the cake needs to be stored cool and cannot be stored in the fridge, I stored it in an alternative location”. However, this would still not inform the user of the temperature outside. We also considered presenting as an explanation the most specific rule, that is, the one that is highest in priority and constitutes an exception to other rules. In the situation presented in Figure 13.3, this would be the rule that items of the cake’s size do not fit in the fridge. In an alternative scenario where the user is aware of the temperature but does not know that the cake is too big to fit in the fridge, this would likely be the appropriate piece of information, but in the situation presented, it is not helpful. In this specific scenario, presenting a fact that strongly influenced the decision (the temperature outside) would indeed be helpful to the user. However, the same fact would not have been helpful to another user who is aware of the temperature but ignorant of the size problem. The reason lies in the fact that it depends entirely on the user which piece of knowledge used by

²We employ defeasible logic, which is a kind of non-monotonic reasoning. Therefore, rules can have priority over other rules if they constitute a more specific exception to a more general rule, e.g., a general rule is “birds can fly” but it is outranked by the exception “penguins are birds but cannot fly”.

the robot is in the uncommon ground.

Suppose that the goal in explaining surprising AI predictions should be to present that part of the agent’s knowledge or reasoning that is part of the uncommon ground. Then, the most salient input features or last-used rules are quite likely shared by the user. This does not help the user discover why the agent acted contrary to their expectation. If the knowledge presented in the explanation is already known to the user, then the user has learned that the agent shares some of their beliefs. However, from their perspective, it should still have arrived at the same conclusion as the user did with this set of beliefs. If, instead, the explanation includes some knowledge or beliefs the user was unaware of or did not share, then the user knows where the uncommon ground is and why the agent arrived at a different conclusion, enabling them to make informed decisions, adapt their behavior or teach the AI new knowledge. We argue that this problem can only be solved by explicitly modeling what the AI agent knows about the user and their previous interactions, which constitute the user and discourse contexts, to provide the agent with a personal and dynamic memory of their interactions. We propose that knowledge from these user-specific contexts can support knowledge in the situational and ontological contexts. If the user is unaware of the temperature outside, then this fact in the situational context does not have any support, as illustrated in Figure 13.4. On the other hand, if the user does not believe in the size problem because they think that an item of this size should fit in the fridge, then this rule in the agent’s ontological context would not have any support. Of course, there will not always be one definite answer to the uncommon ground problem based on the support SUDO model, there can easily be situations where several facts or rules have no support. Similarly, if the user changes their mind, facts that received some support in the past might still be part of the uncommon ground now. Finally, there is not necessarily only one piece of information in the uncommon ground, there can easily be several points of disagreement. However, we propose that our approach presents a viable first step at presenting a helpful explanation tailored to the user, which will give them the necessary information in many circumstances.

At this point, the agent still does not know what the user knows or believes or where precisely the uncommon ground was in this situation: the user has to explicitly instruct or teach the agent for it to gain that knowledge. In the future, users can update the agent’s knowledge base. If they realize that they themselves were missing some correct information, they can update their beliefs and trust the agent’s prediction; or if the difference is not easily resolvable, they might choose not to give the agent this kind of task in the future. Therefore, the generation of the explanation needs to be an extrospective process: it needs to consider what the agent has learned about the user’s beliefs and preferences throughout their previous interactions and then present in the explanation that piece of information that has the lowest support by the user and discourse context. The explanation could include assumptions by the agent that are not explicitly supported by anything it knows about the user, facts that (as far as the agent knows) are not known by the user, or rules that the agent believes but that has never before been used in successful interactions.

Accordingly, extrospective explanations should be evaluated together with prospective users in different contextual scenarios. The goal of good explanations (in situations of surprise) should be to enable the user to understand how the agent’s worldview differs from their own without requiring repeated follow-up questions or presenting superfluous (“obvious”) information to the user. In situations where the user themselves was incorrect or was missing information, they should be able to quickly learn this information from the AI when they ask for an explanation. In personalized AI domains, realizing where the uncommon ground is should empower the user to update the AI’s knowledge base to align with their own preferences if they desire to do so.

13.5 Conclusion

There is a growing awareness in the XAI community that explanations ought to be considered and evaluated in the context of the user for whom they are intended. Previous works have pointed out the problem of implied-but-not-stated contrastive/counterfactual cases when humans ask for explanations [17] as well as that there are many situations in which the user’s goal in asking for an explanation relies on understanding why the AI prediction differed from their expectation [14]. We claim that in the domain of AI in the home (e.g., smart home devices, household robots), extrospective explanations can provide more helpful information for end-users in a more human-centered way, enabling them to understand the reason for unexpected agent behavior. We aim to achieve this by focusing on explanations for transparent, reasoning-based AI systems and explicitly modeling user and discourse context in the agent’s knowledge. If the AI predicts the same outcome based on the same situational knowledge and ontology in two equivalent situations, the respective explanations could differ based on the user asking for them: For each user, the AI would output only a part of their reasoning. Specifically, the AI should output those facts or inferences that it expects are most likely to surprise the user. This should be whatever fact or logical inference the AI has the least reason to believe that the user agrees with or is aware of. In other domains, however, there might be different reasons why explanations should be purely introspective. There are many areas of life where AI is or will be used, in which it will be important that explanations are consistent across different users or that they are a complete representation of the AI reasoning (e.g., medical or legal problems). Therefore, it will be essential to distinguish between explanations aimed at helping individual users understand local, surprising AI predictions and explanations aimed at increasing ML systems’ transparency.

Chapter 13: References

- [EB1] Ashraf Abdul, Jo Vermeulen, Danding Wang, Brian Y. Lim, and Mohan Kankanhalli. Trends and Trajectories for Explainable, Accountable and Intelligible Systems: An HCI Research Agenda. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–18, Montreal QC Canada, April 2018. ACM.
- [EB2] Amina Adadi and Mohammed Berrada. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.
- [EB3] Fatemeh Alizadeh and Dominik Pins. User-friendly Conversational Explanations: A Research Summary. In *Proceedings of the ACM CHI Workshop on ACM CHI Workshop Human-Centered Perspectives in Explainable AI (HCXAI 2022)*, 2022.
- [EB4] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, June 2020.
- [EB5] Michael Braun, Anja Mainz, Ronee Chadowitz, Bastian Pfleging, and Florian Alt. At your service: Designing voice assistant personalities to improve automotive user interfaces. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–11, New York, NY, USA, 2019. ACM.
- [EB6] Herbert H Clark. *Using language*. Cambridge university press, 1996.
- [EB7] Christina Conati, Katrien Verbert, Oswald Barral, and Martijn Millecamp. Personalized XAI for AI-driven Personalization. In *Proceedings of the 2022 Workshop on Explainable Artificial Intelligence (XAI)*, July 2022.
- [EB8] Nils Feldhus, Ajay Madhavan Ravichandran, and Sebastian Möller. Mediators: Conversational Agents Explaining NLP Model Behavior, June 2022. arXiv:2206.06029 [cs].
- [EB9] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining Explanations: An Overview of Interpretability of Machine Learning, February 2019. arXiv:1806.00069 [cs, stat].
- [EB10] Julia Graefe, Selma Paden, and Klaus Bengler. Why and How? A User Study to Evaluate Explainability Within the Automotive Context. In *Proceedings of the ACM CHI Workshop on ACM CHI Workshop Human-Centered Perspectives in Explainable AI (HCXAI 2022)*, 2022.
- [EB11] Andreas Holzinger, Chris Biemann, Constantinos S. Pattichis, and Douglas B. Kell. What do we need to build explainable AI systems for the medical domain?, December 2017. arXiv:1712.09923 [cs, stat].

- [EB12] Lars Kunze, Moritz Tenorth, and Michael Beetz. Putting people’s common sense into knowledge bases of household robots. In *KI 2010: Advances in Artificial Intelligence: 33rd Annual German Conference on AI, Karlsruhe, Germany, September 21-24, 2010. Proceedings 33*, pages 151–159. Springer, 2010.
- [EB13] Thao Le, Tim Miller, Ronal Singh, and Liz Sonenberg. Improving Model Understanding and Trust with Counterfactual Explanations of Model Confidence. In *Proceedings of the IJCAI 2022 workshop on Explainable Artificial Intelligence (XAI)*, 2022. Publisher: arXiv Version Number: 1.
- [EB14] Q. Vera Liao, Daniel Gruen, and Sarah Miller. Questioning the AI: Informing Design Practices for Explainable AI User Experiences. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–15, April 2020. arXiv:2001.02478 [cs].
- [EB15] Silvan Mertes, Christina Karle, Tobias Huber, Katharina Weitz, Ruben Schlagowski, and Elisabeth André. Alterfactual Explanations – The Relevance of Irrelevance for Explaining AI Systems. In *Proceedings of the IJCAI 2022 workshop on Explainable Artificial Intelligence (XAI)*, 2022. Publisher: arXiv Version Number: 1.
- [EB16] Tim Miller. Explanation in Artificial Intelligence: Insights from the Social Sciences, August 2018. arXiv:1706.07269 [cs].
- [EB17] Tim Miller, Piers Howe, and Liz Sonenberg. Explainable AI: Beware of Inmates Running the Asylum Or: How I Learnt to Stop Worrying and Love the Social and Behavioural Sciences, December 2017. arXiv:1712.00547 [cs].
- [EB18] Bamshad Mobasher, Robert Cooley, and Jaideep Srivastava. Automatic personalization based on web usage mining. *Communications of the ACM*, 43(8):142–151, 2000.
- [EB19] Milda Norkute. AI Explainability: Why One Explanation Cannot Fit All. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, Honolulu HI USA, April 2020. ACM.
- [EB20] R. Porzel. *Contextual Computing: Models and Applications*. Cognitive Technologies. Springer Berlin Heidelberg, 2010.
- [EB21] Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller, editors. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*. Springer International Publishing, Cham, 2019.
- [EB22] Rick Tompkins, Ronal Singh, and Tim Miller. The Effect of Diversity in Counterfactual Machine Learning Explanations. In *Proceedings of the IJCAI 2022 workshop on Explainable Artificial Intelligence (XAI)*, 2022.
- [EB23] Maria Wolters, Kallirroi Georgila, Johanna D Moore, Robert H Logie, Sarah E MacPherson, and Matthew Watson. Reducing working memory load in spoken dialogue systems. *Interacting with Computers*, 21(4):276–287, 2009.
- [EB24] Wei Xu. Toward human-centered AI: a perspective from human-computer interaction. *Interactions*, 26(4):42–46, June 2019.
- [EB25] Nima Zargham, Dmitry Alexandrovsky, Jan Erich, Nina Wenig, and Rainer Malaka. “i want it that way”: Exploring users’ customization and personalization preferences for home assistants. In *Extended Abstracts of the 2022 CHI Conference on Human*

Factors in Computing Systems, CHI EA '22, New York, NY, USA, 2022. Association for Computing Machinery.

- [EB26] Nima Zargham, Leon Reicherts, Michael Bonfert, Sarah Theres Voelkel, Johannes Schoening, Rainer Malaka, and Yvonne Rogers. Understanding circumstances for desirable proactive behaviour of voice assistants: The proactivity dilemma. In *Proceedings of the 4th Conference on Conversational User Interfaces*, CUI '22, New York, NY, USA, 2022. Association for Computing Machinery.
- [EB27] Yunfeng Zhang, Q. Vera Liao, and Rachel K. E. Bellamy. Effect of confidence and explanation on accuracy and trust calibration in AI-assisted decision making. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 295–305, Barcelona Spain, January 2020. ACM.
- [EB28] Andreas Zimmermann, Marcus Specht, and Andreas Lorenz. Personalization and context management. *User modeling and user-adapted interaction*, 15(3-4):275–302, 2005.

Chapter 14

Measuring Trust

Rachel Ringe and Robert Porzel

Digital Media Lab, TZI, University of Bremen
Bremen, Germany

Abstract

The success and efficiency of collaborative tasks are heavily influenced by trust, especially in household environments, where mistakes by either party could lead to injuries in addition to failure. With the development of artificial agents for complex household tasks moving forward, it becomes increasingly important to explore the amount of trust put in these autonomous agents and its development. To measure trust in everyday household activities, we propose a metric based on the analysis of delegated tasks during a cooperative everyday activity, such as cooking. We consider the aspects of difficulty, involved risk, possibility of error mitigation as well as supervision by the human to be indicators for the level of trust. We also considered discount factors that might convince a human to delegate a task despite a lack of trust in the autonomous agent. In addition, we present a Virtual Reality test environment that can be used to observe humans during cooperative cooking with an artificial partner together with an evaluation approach for the proposed metric using the test environment.

Keywords: Trust, Metrics, Virtual Reality

14.1 Introduction

Trust is an important factor for the success and effectiveness of a collaborative task in which the participants rely on each other to achieve specific sub-goals[12]. For example, in household environments, such as a kitchen, mistakes can be made by either party that could not only lead to failure to complete the task, but even to injury through various hot or sharp appliances. Trust in a new system or technology is critical to its success, since people tend to employ systems that they trust, and reject systems that they do not trust.

In the last few years, artificial agents, such as vacuuming robots, have become more common in household environments, and assistants for more complex tasks as cooking or cleaning are being developed [1]. To ensure that these new systems will be accepted, it is important to explore how much people trust an autonomous system to handle these tasks, how this trust changes during use and what factors lead to an increase or decrease in trust. Toward that goal,

it is important to find applicable measures for trust. In this paper we propose measuring a user’s trust in an artificial collaborator during cooperative cooking tasks by analysing the tasks delegated to the artificial partner during collaborative execution of a recipe. Therefore, we present a virtual reality (VR) test environment in which experiments measuring these factors can be conducted.

14.2 Related Work

Prior research has focused both on trust in automated systems and specifically robots, as well as on measures of trust for the delegation of tasks to robots. Extensive research has been conducted about the nature of trust and how it influences interactions between humans, as well as between humans and robots and other automated systems. One of the challenges when defining trust in Human-Robot-Interaction is the variety of situations and contexts in which robots are used [5]. Trust in a robot during an evacuation in an emergency scenario would need to be defined differently than trust in a robot during a solely analytical task with no possible danger to the human.

Lee and See define trust as “the attitude that an agent will help achieve an individual’s goals in a situation characterized by uncertainty and vulnerability” [7] while Wagner sees trust as “a belief, held by the trustor, that the trustee will act in a manner that mitigates the trustor’s risk in a situation in which the trustor has put its outcomes at risk” [18]. Both definitions share the notion that one party is dependant on another party to reach their goal, which puts them in a vulnerable and risky position. A large amount of research in human-automation trust suggests that it is mostly performance-based, while trust among humans is additionally influenced by the other agent’s perceived moral values and the relationship between the agents. Malle et al suggest that both of these aspects are part of a multidimensional definition of trust since robots more and more take on the roles of social partners as well [9]. However, the amount of dimensions that apply are dependant on the situation, e.g., that types of robots that foster a certain kind of trust might be more suitable for more trust-dependent scenarios [5].

Often measures of trust are based on self-reports by the study participants where they rate their trust in the robot’s capabilities. These questionnaires vary between the simple question “Do you trust this robot?”, highly situation-specific questionnaires and adjusted psychology questionnaires [3]. Schaefer developed a nuanced scale for trust in human-robot-interaction that can be used in various types of situations [16]. Other possible measures of trust are situation-specific measures, based on user behavior such as:

- following the robot’s guidance in an emergency situation [13]
- degree of the user’s compliance with a robot’s requests [15]
- the frequency of interventions into an autonomous system [19]

These measures of trust can be used to evaluate how different factors influence human trust in robots in research contexts, but also to adapt the behavior of systems at runtime to calibrate user trust to the appropriate level [19] [2].

Trust is a major factor towards the effectiveness of collaboration between robots and humans, especially when it comes to delegation of tasks [4]. Task-delegation during collaborative tasks between humans is influenced by the perceived importance and urgency of the task, as well as the confidence in the abilities of the partner. Novelty of the task, as well as confidence in the ability to mitigate any errors that might arise, play a role in the decision to delegate a task [6]. The term *human supervisory control* describes a relationship between human and automated

system, that mirrors the relationship between human supervisors and their human employees [17]. Task-Delegation interfaces implement a type of supervisory control, and allow humans to delegate tasks of their choosing to automated systems which can then determine the best way to fulfill the given task, and then provide the user with feedback after providing it [11].

The transferability of results, knowledge or skills from a Virtual Reality (VR) environment to the real world and the suitability of VR studies as an alternative to lab studies has been explored in prior research. Mäkela et al found that humans exhibit similar behavior around public displays in a VR environment and the real world equivalent [8]. Paneva et al conducted a study that showed a VR-simulated prototype of a levitation interface yielding similar levels of user experience and engagement as the real prototype [10]. Saffo et al were able to replicate the results of studies on tabletop collaboration and Fitt's law in a VR-environment [14].

Building on this prior work we seek to further research in measuring trust by proposing specific metrics appropriate for the household domain. These proposed metrics will, in turn, be based on the individual task-specific components that are involved in the delegation of the respective tasks.

14.3 Task delegation as Indicator for Trust

As the delegation of tasks among humans relies on trust, we propose that the tasks given to the artificial collaborator, e.g. while preparing a meal, can supply information on the level of trust the human has in them. If the human assigns intricate, dangerous or important tasks to the artificial agent, e.g. heating or cutting an ingredient, this would indicate, that they trust this partner to complete the task successfully. Should they only delegate minor tasks to the robot – for example, wiping the counter – it indicates, that the robotic partner is only trusted to fulfill simple tasks where errors could easily be mitigated.

Toward the goal of measuring trust based on task delegation, three different aspects of a task that could influence a human's tendency to delegate it were chosen in our approach: difficulty, risk and possibility for error mitigation. In addition, it was deemed relevant if a human would supervise the artificial collaborator during a task or not. These aspects were combined into a basis for a scale, that can be used to determine the level of trust the human put into the artificial partner when delegating this specific task to them. The collection of components for the chosen aspects can be seen in Table 14.1 and will be explained in the following section.

Difficulty of the delegated task

Since humans tend to delegate challenging tasks when they have trust in the capabilities of a partner, we assume that the difficulty of a task will be a factor in the amount of trust required for it to be delegated. The first chosen component for difficulty of a task was the amount of different tools or appliances needed for the execution of the task that the artificial partner is meant to complete. This is based on the assumption that tasks that use fewer different tools would be less difficult than ones that use multiple tools/appliances. For example the simple task of wiping the counter-top after use would only require a cloth, while frying mushrooms would require using the stove and a pan as well as a spatula. A second measure for difficulty is the time spent on a task in comparison to all tasks completed during the overall activity. A task that uses only few appliances but takes up half the time collectively spent on a given cooking task is likely more complex than a task that only takes 20 seconds. An example for this would be that mixing a dough for 3 minutes would be more complex than adding oil to the pan. The last measure for difficulty is the cognitive effort, denoted by the number of concepts and

	#	y/n
Difficulty		
Amount of different tools/appliances needed for task		XXX
Time spent on task in relation to all tasks		XXX
Number of terms and axioms needed in ontology		XXX
Risk		
Is the task dangerous to the human?	X	XX
Is the task dangerous to the robot?	X	XX
Possibility of error mitigation		
Is the task irrevocable/non-repeatable?	X	XX
Are the ingredients valuable?	X	XX
Supervision		
Can the task be parallelized and was it parallelized?	X	XX
No supervision by the human	X	XX
No intervention by the human	X	XX
Discount factors		
Is the task boring?	X	XX
Is the task tedious?	X	XX
Is the task unpleasant/gross?	X	XX
Is the user unable to do the task on their own?	X	XX
User-attitude towards the activity		XXX

Table 14.1: Criteria for delegated task that could indicate trust

axioms needed for understanding this task in the used ontology. The higher this number is, the more concepts an agent would need to know and understand to complete the task successfully, hence making it more difficult.

Risk Involved in the Delegated Task

Since everyday activities involve various tools and appliances that might pose a danger to any agent, it is likely that tasks involving these objects would only be delegated to the robot if the human trusts it to not harm humans or itself. A task would be considered dangerous to the human if it involves tools, appliances or ingredients that could cut or burn the human, for example knives or hot water. Tasks that would be considered dangerous to the robot involve liquids that could short-circuit its electronics as well as sharp objects that could damage its components.

Possibility for Error Mitigation

During completion of a household activity there are some tasks where errors are easily mitigated and some where an error would prevent activity from being completed successfully. Delegating task that can not be repeated, e.g., by taking additional ingredients and doing it over or reversed by undoing, would indicate, that the human trusts the robot to complete the task successfully, as it is critical to the completion of the overall activity. In addition, the value of the used ingredients or detergents is included, since a task might be repeatable using additional ingredients, but would involve additional costs. An example would be the use of expensive

spices like saffron or high-grade cleaning fluids.

Supervision during Task Completion

The aspect of supervision during task completion is also pertinent when considering trust, since constant supervision or even intervention likely implies that the human does not trust the robot to complete the task successfully in an autonomous manner. In addition, it is important to consider if the task could be parallelized. If this is a possibility, but the human does not take advantage of this opportunity to speed up the completion of the activity or to increase the efficiency of the process, it could imply, that the human is trying to be ready in case an intervention is needed and thus does not trust the robot to complete the task correctly.

Discount Factors

In addition to the aspects of a task that would imply trust by the human, there were also discount factors included in the criteria, that might convince a human to delegate a task to the robot despite not trusting him to complete it successfully or correctly. A boring or tedious task might be delegated, since the user would rather take the risk of ordering the robot to fulfill it, instead of having to complete it themselves. Examples for this might be peeling a large amount of potatoes or stirring a pot constantly for 30 minutes. Similarly the benefits of delegating a task that the human would consider uncomfortable or disgusting could outweigh the risks of entrusting it to the robot. Another important aspect are the abilities of the human. If the human is incapable to complete a task themselves, they would likely order the robot to do it for them despite not trusting them, since they have no other choice. A possible example for this would be fetching an object from a high shelf, that the human is unable to reach. In addition, the human's attitude towards cooking would be relevant, since it could influence the perceived importance and complexity of tasks greatly. For example someone inexperienced with cooking might consider sauteing onions an easy task and delegate it to the robot without hesitation, while a more experienced cook would consider it far more difficult.

14.4 Test Environment

To observe humans during cooperative cooking with an artificial partner, a VR application was developed in the Unity game engine for use with an Oculus Quest HMD. In this application the user is placed in a kitchen environment together with a virtual robot. The user can interact with various objects in the kitchen by grabbing them with either their hands or the controllers and then complete various cooking tasks by moving them in appropriate ways – e.g., moving a whisk in circular motions through a bowl containing the different ingredients to be mixed. In addition, the user can order the robot to fulfill any of the needed cooking tasks for recipe completion – e.g., portioning a certain amount of an ingredient into a bowl – or some supporting such as cleaning, tidying or fetching objects for the user. For these orders a delegation-type interface is used, where the user orders the robot to fulfill a task in a declarative manner, but is not required to give details on how the task should be completed.

User actions and orders are handled by a highly-accurate physics simulation running in the background of the VR application using the Bullet physics engine, that simulates the correct behavior for objects moved by the user. This involves, for example, that the robot moves autonomously towards objects needed for completion of the order, grasping them and using them in the appropriate ways to complete the task. The robot used in the physics simulation is

abstract and only consists of two grippers and a head. This makes it possible to represent the robot in the VR application with any model that possesses two grippers or similar appendages to interact with objects. An example of this is depicted in Figure 14.1 .

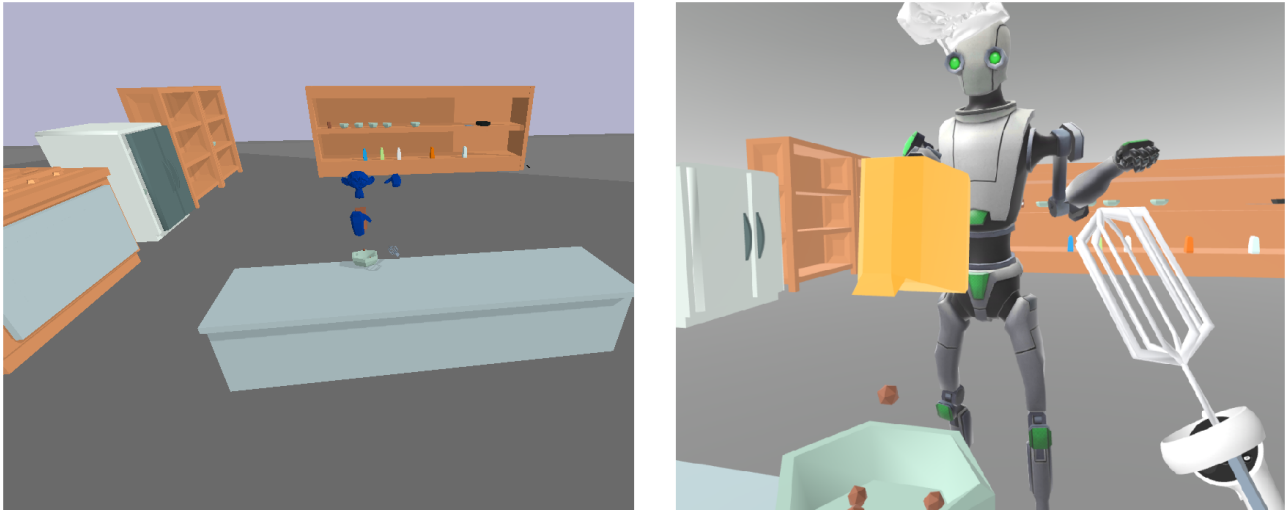


Figure 14.1: Comparison of the physics simulation and the VR application showing the artificial robot portioning an ingredient into a bowl while the user is holding a whisk

During task completion the virtual robot will only make minor mistakes that naturally occur during certain tasks due to the accuracy of the physics simulation and do not influence the result of the action. An example would be spilling a minor amount of an ingredient on the counter top when portioning it into a container. To ensure that the human does not delegate all tasks to the robot or complete all tasks themselves, the task given to the user will include completing the recipe in as little time as possible, thus necessitating some delegation and collaboration.

Evaluation Approach

It is planned to first determine the weights for the different criteria and discount factors described in Section 3 with a user study in the virtual reality test environment, to operationalize the criteria into a scale that rates the trust in the robot displayed by delegating the individual tasks. This data can then be used to implement a graphical model, e.g. a Bayesian network, which could then be used to predict if a human would trust an AI with a certain cooking task. The correctness of these predictions could, subsequently, be tested and updated with further user studies.

14.5 Conclusion

In this work we proposed a scale to rate the trust a human put into an artificial partner during a cooperative household activity. The scale incorporates the difficulty of the task, the risk involved for human and robot, the possibility for error mitigation as well as supervision during task completion. In addition, some discount factors are included, that could influence the human's decision to delegate a task despite insufficient trust in the robot's capabilities. The test environment to conduct experiments using this scale was described as well as an approach to evaluate the results of future experiments.

In the future this scale could be part of a bigger set of measures for trust specific to cooperative tasks, that includes other aspects such as the phrasing of orders given to the robot. Similarly it

could be modified for further household tasks, that could in the future be assigned to household robots. Predictions made by a graphical model based on these metrics could also be used to adjust robot behavior at runtime to calibrate trust to the appropriate level for optimal cooperation. The described test environment and scale could be used in the future to explore different robot appearances and behaviors and how they affect trust, as well as trust development over time when the human can observe the robot complete tasks successfully or make errors.

Chapter 14: References

- [EC1] Michael Beetz, Lorenz Mösenlechner, and Moritz Tenorth. Cram – a cognitive robot abstract machine for everyday manipulation in human environments. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1012–1017. IEEE, 2010.
- [EC2] Min Chen, Stefanos Nikolaidis, Harold Soh, David Hsu, and Siddhartha Srinivasa. Planning with trust for human-robot collaboration. In *Proceedings of the 2018 ACM/IEEE international conference on human-robot interaction*, pages 307–315, 2018.
- [EC3] Meia Chita-Tegmark, Theresa Law, Nicholas Rabb, and Matthias Scheutz. Can you trust your trust measure? In *Proceedings of the 2021 ACM/IEEE international conference on human-robot interaction*, pages 92–100, 2021.
- [EC4] Amos Freedy, Ewart DeVisser, Gershon Weltman, and Nicole Coeyman. Measurement of trust in human-robot collaboration. In *2007 International Symposium on Collaborative Technologies and Systems*, pages 106–114, 2007.
- [EC5] Zahra Rezaei Khavas. A review on trust in human-robot interaction, 2021.
- [EC6] Katherine J Klein, Jonathan C Ziegert, Andrew P Knight, and Yan Xiao. Dynamic delegation: Shared, hierarchical, and deindividualized leadership in extreme action teams. *Administrative science quarterly*, 51(4):590–621, 2006.
- [EC7] John D Lee and Katrina A See. Trust in automation: Designing for appropriate reliance. *Human factors*, 46(1):50–80, 2004.
- [EC8] Ville Mäkelä, Rivu Radiah, Saleh Alsherif, Mohamed Khamis, Chong Xiao, Lisa Borchert, Albrecht Schmidt, and Florian Alt. Virtual field studies: Conducting studies on public displays in virtual reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI ’20, page 1–15, New York, NY, USA, 2020. Association for Computing Machinery.
- [EC9] Bertram F Malle and Daniel Ullman. A multidimensional conception and measure of human-robot trust. In *Trust in human-robot interaction*, pages 3–25. Elsevier, 2021.
- [EC10] Viktorija Paneva, Myroslav Bachynskyi, and Jörg Müller. Levitation simulator: Prototyping ultrasonic levitation interfaces in virtual reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI ’20, page 1–12, New York, NY, USA, 2020. Association for Computing Machinery.
- [EC11] R. Parasuraman, S. Galster, P. Squire, H. Furukawa, and C. Miller. A flexible delegation-type interface enhances system performance in human supervision of multiple robots: empirical studies with roboflag. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 35(4):481–493, 2005.

- [EC12] Rachel Ringe and Robert Porzel. Towards a task-based metric for measuring trust in autonomous robots for everyday activities. In *Proceedings of the CHI TRAIT Workshop on Trust and Reliance in AI-Assisted Tasks*, 2023.
- [EC13] Paul Robinette, Wenchen Li, Robert Allen, Ayanna M. Howard, and Alan R. Wagner. Overtrust of robots in emergency evacuation scenarios. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 101–108, 2016.
- [EC14] David Saffo, Sara Di Bartolomeo, Caglar Yildirim, and Cody Dunne. Remote and collaborative virtual reality experiments via social vr platforms. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2021.
- [EC15] Maha Salem, Gabriella Lakatos, Farshid Amirabdollahian, and Kerstin Dautenhahn. Would you trust a (faulty) robot? effects of error, task type and personality on human-robot cooperation and trust. In *Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction*, pages 141–148, 2015.
- [EC16] Kristin E Schaefer. Measuring trust in human robot interactions: Development of the “trust perception scale-hri”. In *Robust intelligence and trust in autonomous systems*, pages 191–218. Springer, 2016.
- [EC17] Thomas B Sheridan. Human supervisory control. *Handbook of human factors and ergonomics*, pages 990–1015, 2012.
- [EC18] Alan R Wagner and Ronald C Arkin. Recognizing situations that demand trust. In *2011 RO-MAN*, pages 7–14. IEEE, 2011.
- [EC19] Anqi Xu and Gregory Dudek. Optimo: Online probabilistic trust inference model for asymmetric human-robot collaborations. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 221–228, 2015.

Appendix

Initial Kitchen State

Figure 14.2 gives a detailed graphical overview of the initial kitchen state. The ingredients can be found in the freezer, the fridge, and the pantry in the specified quantities. The kitchen cabinet contains all cooking utensils.

Cooking Ontology

The cooking ontology in YAML format can be found in the supplementary materials accompanying this paper. The cooking ontology underlies the procedural semantic representation language, the kitchen states, and both kitchen simulators. As such, it needs to be consulted from both the qualitative simulation, implemented in Common Lisp using Incremental Recruitment Language (IRL), as well as the quantitative simulation, implemented using PyBullet. The YAML format was chosen for the cooking ontology as it is relatively straightforward to edit by hand and parsers for most mainstream programming languages, including Common Lisp and Python, are available. The cooking ontology defined for this benchmark builds further on the SOMA ontology which in itself extends the DUL ontology.

Recipe Corpus

A data dump of the recipe corpus can be found in the supplementary materials accompanying this paper. Here, we briefly explain the data format of the corpus. We chose to use a consistent and structured XML format to represent the recipes in our benchmark. Each recipe XML file consists of four components. First, a unique identifier for the recipe is included. Second, the recipe title is specified. This could be useful as contextual information since it often states the type of dish that should be prepared. Third, the recipe file lists all ingredients that are necessary to prepare the dish. Finally, the recipe file contains a list of instructions that should be executed in order to prepare the dish. An excerpt of the XML format for the Almond Crescent Cookies recipe is shown in Listing 14.1.

The gold-standard annotations for all recipes are found in separate files. These files contain the recipe identifier, followed by one primitive operation per line. However, these annotations are only meaningful as a whole due to common recipe idiosyncrasies, such as missing steps, ellipses and contextual references. An excerpt of the gold-standard annotation for the Almond Crescent Cookies recipe is shown in Listing 14.2.

Cooking Actions

The cooking actions of the procedural semantic representation language, their intended meaning, and simulator-specific implementation choices are listed here in alphabetical order. The number after the name of the cooking action indicates its arity.

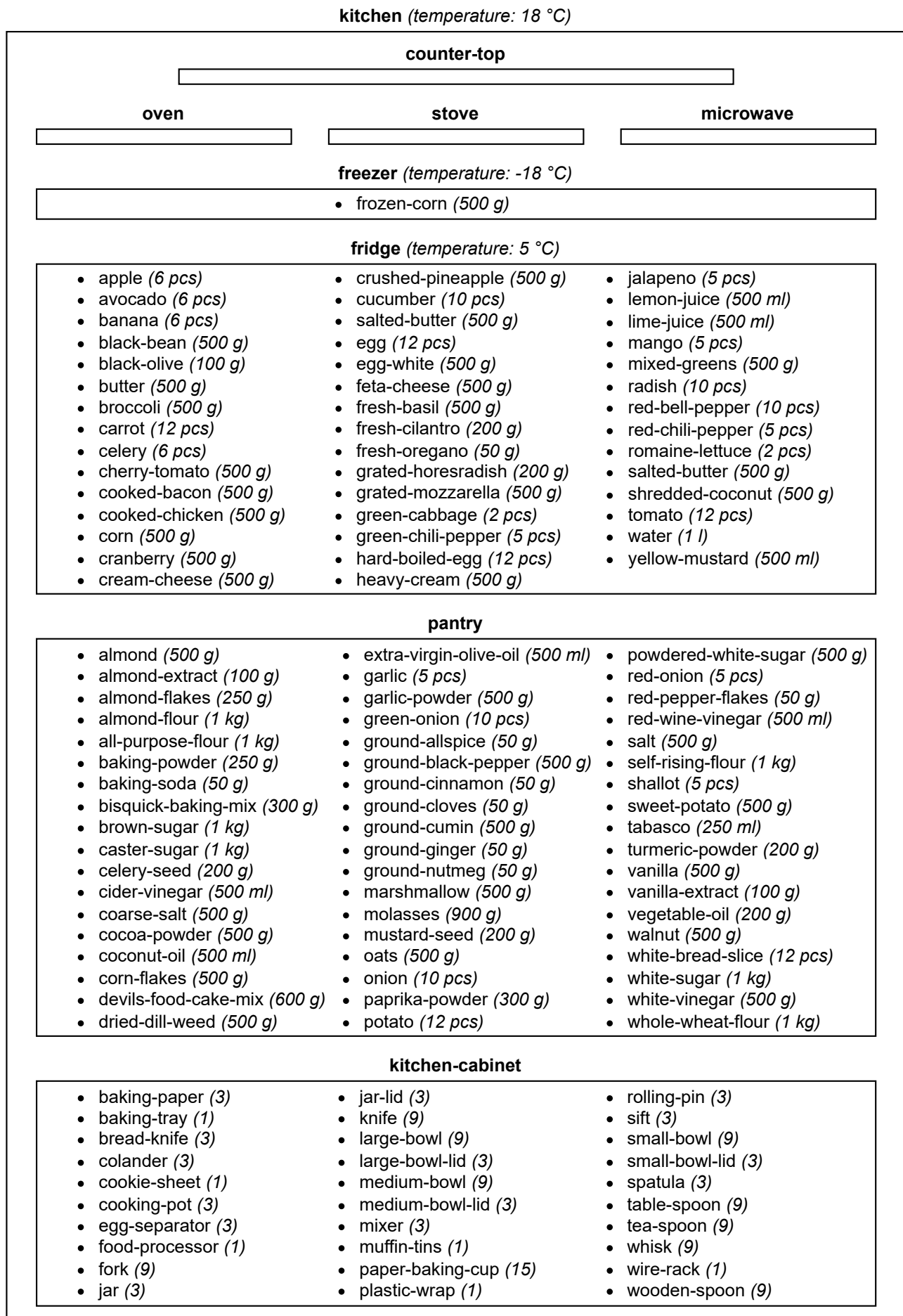


Figure 14.2: Graphical overview of the initial kitchen environment.

Listing 14.1: Excerpt of Almond Crescent Cookies recipe file

```
<recipe>
  <id>almond-crescent-cookies</id>
  <title>Almond Crescent Cookies</title>
  <ingredients>
    <ingredient>
      <utterance>
        230 grams butter, room temperature
      </utterance>
    </ingredient>
    ...
  </ingredients>
  <instructions>
    <instruction>
      <utterance>
        Beat the butter and the sugar together until light and fluffy.
      </utterance>
    </instruction>
    ...
  </instructions>
</recipe>
```

Listing 14.2: Excerpt of Almond Crescent Cookies annotation file

```
#almond-crescent-cookies
(get-kitchen ?kitchen)
(fetch-and-proportion ?proportioned-butter ?ks-with-butter ?kitchen ?target-
  container-1 butter 230 g)
(bring-to-temperature ?warm-butter ?ks-with-warm-butter ?ks-with-butter ?
  proportioned-butter ?room-temp-quantity ?room-temp-unit)
(fetch-and-proportion ?proportioned-sugar ?ks-with-sugar ?ks-with-warm-butter ?
  target-container-2 white-sugar 120 g)
...
```

bake/9

- Arguments:
?baked-thing, ?kitchen-state-out, ?kitchen-state-in, ?thing-to-bake, ?oven, ?time-value, ?time-unit, ?temperature-value, ?temperature-unit
- Intended Meaning:
Obtain *?baked-thing* by baking *?thing-to-bake* in *?oven* at the temperature specified by *?temperature-value* and *?temperature-unit* for the duration specified by *?time-value* and *?time-unit*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?oven* defaults to the closest unused oven in the kitchen
 - *?temperature-value* and *?temperature-unit* default to the temperature of *?oven* (only possible in case *?oven* is specified)
- Constant Arguments:
 - *?time-value* and *?temperature-value* must be numerical values
 - *?time-unit* must be *hour* or *minute*
 - *?temperature-unit* must be *degrees-celsius*

boil/8

- Arguments:
?boiled-thing, ?kitchen-state-out, ?kitchen-state-in, ?thing-to-boil, ?stove, ?heating-setting, ?time-value, ?time-unit
- Intended Meaning:
Obtain *?boiled-thing* by boiling *?thing-to-boil* on the *?stove* at the heating setting specified by *?heating-setting* for the duration specified by *?time-value* and *?time-unit*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?stove* defaults to the closest unused stove in the kitchen
 - *?heating-setting* defaults to *medium-heat*
 - *?time-value* and *?time-unit* default to 30 minutes
- Constant Arguments:
 - *?heating-setting* must be *low-heat, medium-heat, medium-high-heat, high-heat*
 - *?time-value* and *?temperature-value* must be numerical values
 - *?time-unit* must be *hour* or *minute*

beat/5

- Arguments:
?beaten-thing, ?kitchen-state-out, ?kitchen-state-in, ?thing-to-beat, ?beating-tool

- Intended Meaning:
Obtain *?beaten-thing* by beating *?thing-to-beat* using *?beating-tool*. Beating can be seen as a more intense form of mixing which adds some air bubbles during the combination process. This form of mixing also leads to a homogeneous result as individual components are not kept intact. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?beating-tool* defaults to the closest unused whisk in the kitchen

bring-to-temperature/6

- Arguments:
?thing-at-desired-temperature, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-bring-to-temperature*, *?temperature-value*, *?temperature-unit*
- Intended Meaning:
Obtain *?thing-at-desired-temperature* at the temperature specified by *?temperature-value* and *?temperature-unit* by waiting for *?thing-to-bring-to-temperature* to cool off or warm up by advancing towards the ambient temperature. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?temperature-value* and *?temperature-unit* default to the current room temperature of the kitchen, which is around 18 °C.
- Constant Arguments:
 - *?temperature-value* must be a numerical value
 - *?temperature-unit* must be *degrees-celsius*

cover/5

- Arguments:
?covered-thing, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-cover*, *?cover*
- Intended Meaning:
Obtain *?covered-thing* by covering *?thing-to-cover* with the specified *?cover*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?cover* defaults to an appropriate cover for the given *?thing-to-cover*, i.e., a *bowl-lid* for a *bowl*, a *jar-lid* for a *jar* or *plastic-wrap* for anything else.

cut/7

- Arguments:
?cut-thing, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-cut*, *?cutting-pattern*, *?cutting-tool*, *?cutting-surface*

- Intended Meaning:
Obtain *?cut-thing* by using *?cutting-tool* to cut *?thing-to-cut* on the *?cutting-surface* according to the specified *?cutting-pattern*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?cutting-tool* defaults to the closest unused knife in the kitchen
 - *?cutting-surface* defaults to the closest unused cutting board in the kitchen
- Constant Arguments:
 - *?cutting-pattern* must be *chopped*, *finely-chopped*, *slices*, *fine-slices*, *squares*, *two-cm-cubes*, *halved*, *shredded*, *minced*, or *diced*

crack/5

- Arguments:
?container-with-whole-eggs, *?kitchen-state-out*, *?kitchen-state-in*, *?eggs-to-crack*, *?target-container-for-whole-eggs*
- Intended Meaning:
Obtain *?container-with-whole-eggs* by cracking *?eggs-to-crack*, i.e., removing the egg shell from *?eggs-to-crack*, and dropping the egg contents in the container specified by *?target-container-for-whole-eggs*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?target-container-for-whole-eggs* defaults to the closest unused medium bowl in the kitchen

dip/5

- Arguments:
?dipped-thing, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-dip*, *?dip*
- Intended Meaning:
Obtain *?dipped-thing* by dipping *?thing-to-dip* into *?dip*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.

drain/6

- Arguments:
?drained-thing, *?remaining-liquid*, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-drain*, *?draining-tool*
- Intended Meaning:
Obtain *?drained-thing* by draining *?thing-to-drain* using *?draining-tool* leaving the remaining liquid in *?remaining-liquid*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?draining-tool* defaults to the closest unused colander in the kitchen

fetch/5

- Arguments:
?fetched-thing, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-fetch*, *?quantity-to-fetch*
- Intended Meaning:
Obtain *?fetched-thing* by locating one or more *?thing-to-fetch* objects in the kitchen and bringing it to a common work area such as a kitchen countertop. The exact number of objects to fetch is specified by *?quantity-to-fetch*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Constant Arguments:
 - *?quantity-to-fetch* must be a numerical value
 - *?thing-to-fetch* must be any transferable container or cooking utensil available in the kitchen environment.

fetch-and-proportion/7

- Arguments:
?fetched-and-proportioned-ingredient, *?kitchen-state-out*, *?kitchen-state-in*, *?target-container-for-proportioned-ingredient*, *?ingredient-to-fetch-and-proportion*, *?proportion-value*, *?proportion-unit*
- Intended Meaning:
Obtain an amount of the food product *?fetched-and-proportioned-ingredient* by fetching an *?ingredient-to-fetch-and-proportion*, taking a portion from it specified by *?proportion-value* and *?proportion-unit* and placing this portion inside the container specified by *?target-container-for-proportioned-ingredient*. Ingredient leftovers are returned to their original location. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Constant Arguments:
 - *?ingredient-to-fetch-and-proportion* must be any ingredient that is mentioned in the ingredient list of a supported recipe. Ingredients should be specified by replacing all spaces in an ingredient name with the minus sign (-), e.g., ‘ground black pepper’ would become *ground-black-pepper*.
 - *?proportion-value* must be a numerical value
 - *?proportion-unit* must be *piece*, *g*, *teaspoon*, *tablespoon*, *l* or *ml*

flatten/5

- Arguments:
?flattened-thing, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-flatten*, *?flattening-tool*
- Intended Meaning:
Obtain *?flattened-thing* by flattening *?thing-to-flatten* using *?flattening-tool*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:

- *?flattening-tool* defaults to the closest unused rolling pin in the kitchen

flour/5

- Arguments:
?floured-thing, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-flour*, *?flour*
- Intended Meaning:
Obtain *?floured-thing* by flouring *?thing-to-flour* with *?flour*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?flour* defaults to 10 grams of all-purpose flour taken from the closest container with all-purpose flour

fry/8

- Arguments:
?fried-thing, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-fry*, *?stove*, *?heating-setting*, *?time-value*, *?time-unit*
- Intended Meaning:
Obtain *?fried-thing* by frying *?thing-to-fry* on the *?stove* at the heating setting specified by *?heating-setting* for the duration specified by *?time-value* and *?time-unit*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?stove* defaults to the closest unused stove in the kitchen
 - *?heating-setting* defaults to *medium-heat*
 - *?time-value* and *?time-unit* default to 30 minutes
- Constant Arguments:
 - *?heating-setting* must be *low-heat*, *medium-heat*, *medium-high-heat*, *high-heat*
 - *?time-value* and *?temperature-value* must be numerical values
 - *?time-unit* must be *hour* or *minute*

get-kitchen/1

- Arguments:
?initial-kitchen-state
- Intended Meaning:
Obtain the initial state of the kitchen *?initial-kitchen-state*. This is expected to provide access to an environment model of the kitchen to provide contextual information needed for executing a recipe.
- Default Values:

- *?initial-kitchen-state* defaults to the initial kitchen state in which a recipe will be executed. This argument is expected to be left to its default value in which case this primitive functions as a ‘getter’.

grease/5

- Arguments:
?greased-thing, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-grease*, *?thing-to-grease*, *?grease*
- Intended Meaning:
Obtain *?greased-thing* by greasing *?thing-to-grease* with *?grease*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?grease* defaults to 10 grams of butter taken from the closest container with butter

grind/5

- Arguments:
?ground-thing, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-grind*, *?grinding-tool*
- Intended Meaning:
Obtain *?ground-thing* by grinding *?thing-to-grind* using *?grinding-tool*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?grinding-tool* defaults to the closest unused food-processor in the kitchen

leave-for-time/6

- Arguments:
?cooled-thing, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-cool*, *?time-value*, *?time-unit*
- Intended Meaning:
Obtain *?cooled-thing* by waiting for the duration specified by *?time-value* and *?time-unit* to let *?thing-to-cool* cool off towards the ambient temperature. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Constant Arguments:
 - *?time-value* and *?temperature-value* must be numerical values
 - *?time-unit* must be *hour* or *minute*

line/5

- Arguments:
?lined-thing, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-line*, *?lining*
- Intended Meaning:
Obtain *?lined-thing* by lining *?thing-to-line* with *?lining*, e.g., lining a baking tray with

some baking paper or lining muffin tins with paper baking cups. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.

- Default Values:
 - *?lining* defaults to the closest unused sheet of baking paper
- Constant Arguments:
 - *?lining* must be *baking-paper* or *paper-baking-cups*
 - *?thing-to-line* must be *baking-tray*, *cookie-sheet*, *pan* or *muffin-tins*

mash/5

- Arguments:
?mashed-thing, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-mash*, *?mashing-tool*
- Intended Meaning:
Obtain *?mashed-thing* by mashing up *?thing-to-mash* using *?mashing-tool*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?mashing-tool* defaults to the closest unused fork in the kitchen

melt/5

- Arguments:
?melted-thing, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-melt*, *?melting-tool*
- Intended Meaning:
Obtain *?melted-thing* by melting *?thing-to-melt* using *?melting-tool*. This melting tool could be any kind of heating appliance in the kitchen, ranging from a pan on the stove to a microwave. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?melting-tool* defaults to the closest unused microwave in the kitchen

mingle/5

- Arguments:
?mingled-thing, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-mingle*, *?mingling-tool*
- Intended Meaning:
Obtain *?mingled-thing* by mingling *?thing-to-mingle* using *?mingling-tool*. Mingling can be seen as a softer form of mixing in which the individual components are still kept intact during the combination process. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?mingling-tool* defaults to the closest unused wooden spoon in the kitchen

mix/5

- Arguments:
?mixed-thing, ?kitchen-state-out, ?kitchen-state-in, ?thing-to-mix, ?mixing-tool
- Intended Meaning:
Obtain *?mixed-thing* by mixing *?thing-to-mix* using *?mixing-tool*. Mixing can be seen as a form of mixing that is intense enough to achieve a homogeneous mixture without being so intense that air bubbles are added during the mixing process. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?mixing-tool* defaults to the closest unused whisk in the kitchen

peel/6

- Arguments:
?peeled-thing, ?peel, ?kitchen-state-out ?kitchen-state-in, ?thing-to-peel, ?peeling-tool
- Intended Meaning:
Obtain *?peeled-thing* and its *?peel* by peeling *?thing-to-peel* using *?peeling-tool*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?peeling-tool* defaults to the closest unused knife in the kitchen

portion-and-arrange/8

- Arguments:
?portions, ?kitchen-state-out, ?kitchen-state-in, ?thing-to-portion, ?portion-size-value, ?portion-size-unit, ?placement-pattern, ?container-for-portions
- Intended Meaning:
Obtain *?portions* by portioning *?thing-to-portion* into portions that each have a size specified by *?portion-size-value* and *?portion-size-unit*. These portions are placed onto the container *?container-for-portions* following the *?placement-pattern*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?placement-pattern* defaults to a pattern in which all portions are evenly spread out over the container
 - *?container-for-portions* defaults to the countertop of the kitchen
 - *?portion-size-value* and *?portion-size-unit* default to portion sizes that cause an equal division over the available tins (only possible in case *?container-for-portions* are muffin tins)
- Constant Arguments:
 - *?placement-pattern* must be *side-to-side*, *evenly-spread*, or *5-cm-apart*

preheat-oven/6

- Arguments:
?preheated-oven, ?kitchen-state-out, ?kitchen-state-in, ?oven, ?temperature-value, ?temperature-unit
- Intended Meaning:
Obtain *?preheated-oven* by changing the settings of the *?oven* to reach the temperature specified by *?temperature-value* and *?temperature-unit*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?oven* defaults to the closest unused oven in the kitchen
- Constant Arguments:
 - *?temperature-value* must be a numerical value
 - *?temperature-unit* must be *degrees-celsius*

refrigerate/7

- Arguments:
?refrigerated-thing, ?kitchen-state-out, ?kitchen-state-in, ?thing-to-refrigerate, ?refrigerator, ?time-value, ?time-unit
- Intended Meaning:
Obtain *?refrigerated-thing* by putting *?thing-to-refrigerate* inside *?refrigerator* for the duration specified by *?time-value* and *?time-unit*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?refrigerator* defaults to the closest unused fridge in the kitchen
 - *?time-value* and *?time-unit* default to one hour
- Constant Arguments:
 - *?time-value* must be a numerical value
 - *?time-unit* must be *minute* or *hour*

seed/6

- Arguments:
?seeded-thing, ?seed, ?kitchen-state-out, ?kitchen-state-in, ?thing-to-seed, ?seeded-tool
- Intended Meaning:
Obtain *?seeded-thing* and its *?seed* by seeding *?thing-to-seed* using *?seeding-tool*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?seeding-tool* defaults to the closest unused knife in the kitchen

separate-eggs/8

- Arguments:
?egg-yolks, *egg-whites*, *?kitchen-state-out*, *?kitchen-state-in*, *?eggs*, *?container-for-yolks*, *?container-for-whites*, *?egg-separator*
- Intended Meaning:
Obtain *?egg-yolks* and *egg-whites* by using an *?egg-separator* to separate separating the whole *?eggs* into the *?container-for-yolks* and *?container-for-whites* respectively. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?container-for-yolks* defaults to the closest unused stove in the kitchen
 - *?container-for-whites* defaults to the closest unused medium bowl in the kitchen (excluding the one found for *?container-for-yolks*)
 - *?egg-separator* defaults the closest unused egg separator in the kitchen

shake/4

- Arguments:
?shaken-thing, *?kitchen-state-out* *?kitchen-state-in*, *?thing-to-shake*
- Intended Meaning:
Obtain *?shaken-thing* by shaking *?thing-to-shake* to mix its contents, which are generally liquids, until a homogeneous mixture is reached. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.

shape/5

- Arguments:
?shaped-thing, *?kitchen-state-out* *?kitchen-state-in*, *?thing-to-shape*, *?shape*
- Intended Meaning:
Obtain *?shaped-thing* by shaping *?thing-to-shape* into the shape specified by *?shape*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Constant Arguments:
 - *?shape* must be *ball-shape* or *crescent-shape*

sift/6

- Arguments:
?sifted-thing, *?kitchen-state-out* *?kitchen-state-in*, *?container-to-sift-into*, *?thing-to-sift*, *?sift*
- Intended Meaning:
Obtain *?sifted-thing* by using *?sift* to sift *?thing-to-sift* into the container specified by *?container-to-sift-into*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.

- Default Values:
 - *?container-to-sift-into* defaults to the closest unused large bowl in the kitchen
 - *?sift* defaults to the closest unused sift in the kitchen

spread/6

- Arguments:
?thing-with-spread-on, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-spread-on*, *?thing-to-spread*, *?spreading-tool*
- Intended Meaning:
Obtain *?thing-with-spread* by spreading *?thing-to-spread* on *?thing-to-spread-on* using *spreading-tool*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?spreading-tool* defaults to the closest unused spatula in the kitchen

sprinkle/5

- Arguments:
?thing-with-sprinkles-on, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-sprinkle-on*, *?sprinkles*
- Intended Meaning:
Obtain *?thing-with-sprinkles-on* by sprinkling *?sprinkles* onto *?thing-to-sprinkle-on*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.

transfer-contents/8

- Arguments:
?container-with-transferred-contents, *?container-with-rest-of-contents*, *?kitchen-state-out*, *?kitchen-state-in*, *?container-to-transfer-contents-to*, *?container-with-contents-to-transfer*, *?value-of-transfer-amount*, *?unit-of-transfer-amount*
- Intended Meaning:
Obtain *?container-with-transferred-contents* and *?container-with-rest-of-contents* by transferring an amount (specified by *?value-of-transfer-amount* and *?unit-of-transfer-amount*) of the container *?container-with-contents-to-transfer*'s contents into *?container-to-transfer-contents-to* leaving the remaining contents in *?container-with-rest-of-contents*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?container-to-transfer-contents-to* defaults to the closest unused large bowl in the kitchen
 - *value-of-transfer-amount* and *unit-of-transfer-amount* default to an amount for which all contents are transferred, effectively emptying the original container
- Constant Arguments:

- *?value-of-transfer-amount* must be a numerical value
- *?unit-of-transfer-amount* must be *piece*, *g*, *teaspoon*, *tablespoon*, *ml*, or *percent*

transfer-items/6

- Arguments:
?transferred-items, *?kitchen-state-out*, *?kitchen-state-in*, *?items-to-transfer*, *?placement-pattern*, *?destination*
- Intended Meaning:
Obtain *?transferred-items* by carefully transferring all items from *?items-to-transfer* to *?destination* and placing them there according to the pattern specified by *?placement-pattern*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.
- Default Values:
 - *?placement-pattern* defaults to a pattern in which the available location is filled up from side to side by creating rows of items one at a time in which items are placed next to each other.

uncover/5

- Arguments:
?uncovered-thing, *?cover* *?kitchen-state-out*, *?kitchen-state-in*, *?covered-thing*
- Intended Meaning:
Obtain *?uncovered-thing* and its prior *?cover* by removing the cover from *?covered-thing*. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.

wash/4

- Arguments:
?washed-thing, *?kitchen-state-out*, *?kitchen-state-in*, *?thing-to-wash*
- Intended Meaning:
Obtain *?washed-thing* by rinsing off or washing *?thing-to-wash* with water. The arguments *?kitchen-state-in* and *?kitchen-state-out* represent the contextual situation before and after execution of this predicate.