Aspects of Application of Neural Recognition to Digital Editions

ABSTRACT – Artificial neuronal networks (ANN) are widely used in software systems which require to solve problems without a traditional algorithmic approach, like in character recognition: ANN learn by example, so that they require a consistent and well-chosen set of samples to be trained to recognize their patterns. The network is thaught to react with high activity in some of its output neurons whenever an input sample belonging to a specified class (e.g. a letter shape) is presented to it, and has the capability of assessing the similarity of samples never encountered before to any of these models. Typical OCR applications thus require a significant amount of preprocessing for such samples, like resizing images and remove all the 'noise' data letting the letter countours emerge clearly from the background. Further, usually a huge number of samples is required to effectively train a network to recognize a character against all the others. This may represent an issue for paleographical applications, because of the relatively low quantity and high complexity of digital samples available, and poses even more problems when our aim is detecting subtle differences (e.g. the special shape of a specific letter from a well-defined period and *scriptorium*). It would be probably wiser for scholars to define some guidelines for extracting from samples the features defined as most relevant according to their purposes, and let the network to deal with just a subset of the overwhelming amount of detailed nuances available. ANN are no magic, and it is always the careful judgement of scholars to provide a theorical foundation for any computer-based tool they might want to use to help them solve their problems: we can easily illustrate this point with samples drawn from any other application of IT to humanities. Just as we can expect no magic in detecting alliterations in a text if we just feed a system with a bunch of letters, we can no more claim that a neural recognition system might be able to perform well with a relatively small sample where each shape is fed as it is, without instructing the system about the features scholars define as relevant. Even before ANN implementations, it is right this theorical background which must be put into test when planning such systems.

Scenario

This paper derives from a much wider discussion of a digital edition system[1] I have been creating for collecting a large amount of different data related to a specific subject. One of the chief points I often stress about the principles of this system is that its purpose is creating a 'truly' digital edition of textual and/or non-textual material, which should also be treated as a research tool rather than as a mere clone of a traditional paper edition. Also, the system aims to be greatly expandable so that new, often very specialized data can be added to the existing material at any time, attaching to its structure rather than modifying it. For instance, a collection of epigraphical texts can be enriched by specialized data about linguistics, metrics, history, prosopography, archaeology, paleography, etc. which may eventually come from expert systems capable of providing automated analysis (like in the case of metrical analysis); also, some of the existing data, like e.g. a collection of digital drawings taken from photographs, can also lay out the foundation for further developments which may sound particularly interesting in fields like paleography. Here I'd just like to discuss some basic aspects of one of these suggested applications, exploring the feasibility of applying neural techniques to graphical data: among others, a further application for the multimedia capabilities of digital editions can be provided by patterns recognition using neural methods. Typically we can start with the sample of inscriptions, even if such methods can be applied to several other scenarios (e.g. manuscripts, brick stamps, iconography, coins, etc.), probably with different levels of efficiency and different requirements for data preparation. In the hypothetical scenario of an epigraphical edition already provided with several digital drawings extracted from their photographs, the next step might be extracting the shapes of each letter from each of them, and then feed a neural recognition system with them as samples for a given period or region.

[1] For what follows see e.g. Fusi 2007 (principles for the epigraphical system) and Fusi 2008 (expert metrical system), together with their hosting website.

digital drawing

some shapes extracted from the drawing

For instance, we could extract all the letters forms from selected Greek inscriptions in our corpus (see the pictures above) and use them to train a neural system providing it with different chronological classes. Once this is done, we could imagine a publishing scenario where users perform truly graphical query to retrieve all the inscriptions whose letters resemble a provided sample, which might come from a photo, or from a drawing sketched by the user himself with an inking system; such a system might then provide a raw hint for the datation of the inscription the letter sample has been taken from, or just any other type of paleographical classification we have defined as relevant in our texts. A by-product of this system (or rather a prerequisite for it) might also be a detailed catalogue listing all the shapes of all the paleographically relevant inscriptions in a corpus, where users can browse them on screen century by century, or according to any other filtering or sorting criterion, and immediately get a feeling of the evolution of the writing system. This is just a small example but the possibilities might be endless; here I'd just like to stress the importance of fully exploiting the capabilities of a true digital edition; automatic pattern recognition is just another sample of this statement.

Pattern Recognition and ANN

In today's software, pattern recognition (including of course optical character recognition, OCR) is typically accomplished using artificial neural networks (ANN). ANN are a complex subject, and even an introduction to them would be out of the scope of this work (and far away the competence of the writer), so that here I'll just sketch some of their principles with reference to their practical application to software systems[2]. The easiest way of grasping the way an ANN works is comparing its behavior with traditional software solutions, which typically use an algorithmic approach for solving problems: there is a sequence of instructions to follo, and this of course implies that we must know them in advance. Neural networks instead, composed by interconnected elements (neurones) working in parallel, learn by example, so that they cannot be programmed to perform a specific task. This implies that selecting the right examples and defining the relevance of their traits is a crucial point.

An artificial neural network is a set of nodes and connections between them; the nodes (neurons) are the computational units: they get some input and process them to produce an output. It is right the interaction of nodes through their connections which leads to define an emergent behaviour for the network, so that its abilities supercede the ones of its elements. These artificial neurons are inspired by natural neurons, which receive signals through synapses; when the strength of the signals exceeds a certain treshold the neuron is activated and emits a signal through the axon. This signal in turn might be sent to another synapse, and might activate other neurons, and so on. An artificial neuron is a device with several inputs and a single output. These inputs are multiplied by weights (strength of the respective signals) and then computed by mathematical functions which determine if a neuron should be activated or not. Another function, which may or not be different, calculates the output of the neuron, often related to a specific treshold. By adjusting the weights of a neuron we can obtain the output we desire for each specific input; as a typical ANN contains many neurons, it would be very complex to do such adjustments by hand, so special algorithms are used; this process of adjusting weights represents the training or learning of the ANN. A neuron thus operates in training mode or in using mode: in the training mode it is trained to fire or not for particular input patterns, i.e. to associate a specific input to a specific output. In using mode a neuron either encounters a taught pattern or a new one; in the former case it simply outputs fires the taught output, in the latter case a firing rule is used to determine whether to fire or not.

In connection to our subject it is to be stressed that a firing rule relates not only to the taught input patterns, but to all the input patterns. For instance, a firing rule can be implemented with a technique which compares elements in patterns so that the neuron fires or not according to whether the compared patterns have more input elements in common with

[2] Introductions to ANN abund in the web. Here I'll mainly follow (with consistent semplifications) Stergiou & Siganos and Gershenson. Other material can be found in the documentation of the API of several open-source and/or freeware software libraries implementing ANN or their derivatives (see e.g. Chang & Lin LIBSVM website for a list of ports of this software library in several languages).

the nearest pattern in the firing-taught set or in the not-firing-taught set. Such a rule provides the neuron with the 'sense' of similarity so that it can respond to patterns never encountered before, returning the output corresponding to the taught input pattern which is most similar to the given pattern. Also, more complicated neurons can be used, where inputs are 'weighted' by multiplying the input values of a pattern by a specific number and adding their results, letting the neuron fire only when the sum exceeds a predefined treshold value. Such neurons can adapt to specific situations by changing their weights or treshold; there are several algorithms for adapting, and one of the commonest is the back error propagation, where we start with randomly chosen weights and then adjust them so that the error is minimal.

A very common type of ANN is built of three sets of neurons: a layer of input neurons is connected to a middle layer of "hidden" neurons, which in turn is connected to a layer of output neurons. The activity in the input layer represents the input data fed into the network; the activity of the middle layer is defined by the activity of the input layer and the weights assigned to the connections between the input and the middle layer; and in turn the activity of the output layer is defined by the activity of the middle layer and the weights assigned to the connections between the middle and the output layer. In this model the middle layer is thus free to build its own representation of the input, by connecting one or more neurons of the input layer to one or more neurons of the middle layer and assigning various weights to these connections. The input pattern units can thus be variously grouped, combined, selected and weighted according to the specific problem the ANN should solve.

These connections and their weights define the 'knowlegde' of an ANN network; modifying this knowledge according to experience implies a learning rule which can change these weights. Typically for pattern recognition the learning is supervised, in the sense that it happens by associating input samples (e.g. the various shapes of the character A) with their desired output (the character identified as A). For instance, say we want to recognize the 26 printed lowercase letters of the standard ASCII alphabet (a-z): we might use a grid of 16×16 optical sensors, each capable of detecting the presence or absence of ink on a small portion of the printed character area. We would thus need an input layer of 256 units (16×16) and an output layer of 26 units (one for each letter): for each letter the network should produce high activity in the corresponding output neuron and low activities in all the other output neurons. For instance, for a correctly recognized letter c there might also be a fair amount of activity in the output neuron corresponding to the letter o, which "looks" like c, but the activity in the output neuron corresponding to o should be considerably higher. To train the network we would present it a set of sample images for each letter, compare the activity produced in the output layer by each of them and calculate the error (defined as the square of the difference between the actual and the desired activities). We would then use some algorithm to adjust the weights of each connection in order to reduce the error at its minimum and thus get the best recognition results.

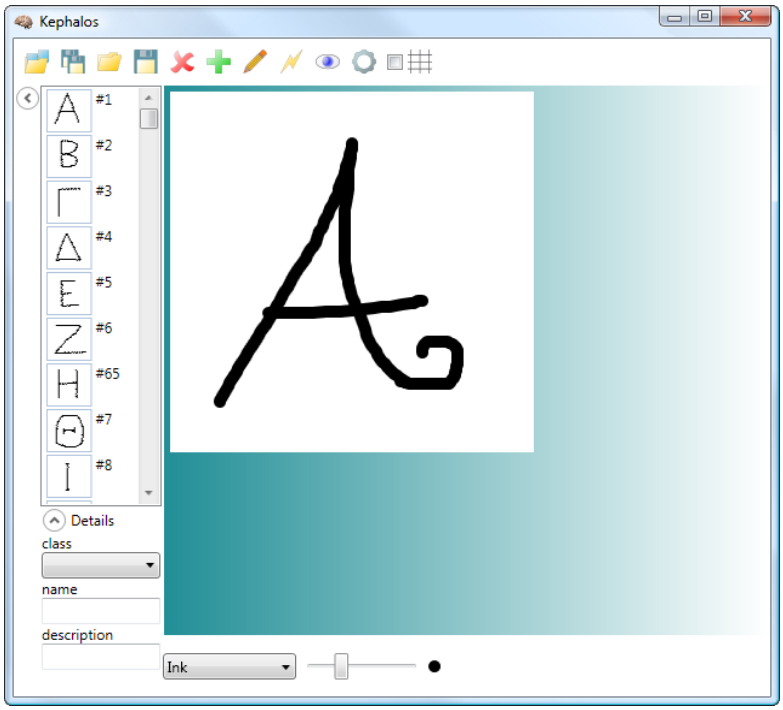
As shown in this sample we are thus representing each letter shape with a set of units, i.e. a vector of numerical values: just think of superposing an ideal grid to a printed character, and mapping each grid cell to a number representing the presence (e.g. 1) or absence (e.g. 0) of ink on a small portion of the printed character area. For instance, if we were using a 3x5 grid we might represent a letter A like:

	X	
X		X
X	X	X
X		X
X		X

which in turn might be represented by a vector of numbers like [0,1,0, 1,0,1, 1,1,1, 1,0,1, 1,0,1]. Of course, in a generic OCR scenario a first issue is represented by the amount of preprocessing required to extract such a black and white silouhette from a photographic image: first of all we must reduce the amount of non-essential information, which for a digital image typically means resizing each character image to a predefined size using the proper resampling algorithm, removing color information, reducing a grayscale image to a black and white one and applying any adjustment technique considered useful for letting the image contours emerge clearly from the background 'noise'. This step can be very complex and imply a lot of digital image processing techniques. Also, the size of the character should be accurately chosen so that it is not too small (otherwise complex shapes might result into a barely readable black spot once resized) nor too big (which might result into vectors too big to be manipulated in learning and recognition). Nevertheless, it might also happen that our chosen vectors size is still impacting the performance of our network: indeed, it's easy for a vector size to grow when representing letters, especially when their nuances are essential. Even a very small grid like 16×16, which would probably be too small to be usable for paleographical purposes, implies 256-units vectors, and just doubling the grid would already take us to 1024 units. Computers are fast, but if you think of the complexity and number of calculations implied by the usage of an ANN built of at least 3 interconnected layers with several input characters each represented by thousands of units, which must be trained adjusting even hundreds of times the weights of the whole system it is easy to understand that we might incur in very serious performance problems. To this end, OCR specialists often use additional processing techniques like receptors: for instance, think of a grid of points representing a character shape; instead of using vectors containing a unit for each point we might imaginarily draw a set of line segments over the shape, with arbitrary size and directions, and state that a receptor has an activated value when

it crosses a letter and a deactivated value when this does not happen. This way we would just have as many units per vector as receptors (i.e. lines). This technique (which of course implies the problem of generating such receptors in an efficient way) may be very powerful in some contexts, where we can be satisfied in detecting the essential traits of each character in order to distinguish them, but it does not fit well where our shapes are rather complex and our objective is to pay attention to finer nuances.

For such applications it is probably better to consider a relatively recent outgrowth of ANN, support vector machines (SVM), which are close to ANN but typically outperform it when dealing with specific problems like pattern recognition; SVM work by finding the optimal 'boundary' ("hyperplane") which divides groups of vectors (i.e. the set of features which define a pattern) so that (all or most of) the ones belonging to a recognized class (e.g. a letter, when recognizing characters) stand on one side of the plane, while all the others stand on the other side (the vectors along this 'boundary' are named support vectors, whence the term). As for ANN, there are several open-source libraries available for building applications using such techniques; a sample of a trivial application based onto one of these libraries (built in C# and just slightly modified by myself for easier use in an asynchronous environment) is represented by the demo shown below:



In this demo application, users literally draw a set of shapes (which might represent letters or any other thing), assign them a class (e.g. tell the software that the given sample represents an A, another a B, etc.), and let the system learn from them by getting its own 'idea' of each of the declared classes. Once the system is trained the user can draw a new shape and ask to identify it.

Application Issues

The above sample is just a demo, but for its very nature this is a field where it's very difficult to make previsions on the efficiency of any given real-world system before actually putting it into test. From a practical standpoint, detecting patterns using such methods poses several problems. First of all, as we have seen all these neural approaches require very careful training and consequently a very high number of samples (i.e. images, in our case): these techniques solve problems by learning to associate (in various complex ways) input patterns (e.g. a given shape for letter A) with output patterns (the letter A in itself): there is no algorithmic approach here, as (fortunately!) we're not going to tell the machine which are all the steps to recognize each shape as a specific letter. We just 'show' it samples for each letter, and let it 'figure' out to which of these samples any given sample is nearest. This is of course the only way of handling such problems rather than thinking exclusively in algorithmic terms, which might easily turn the desired solution into a programmer's nightmare, even supposing that we can imagine a set of well-defined and ordered steps to instruct a machine to distinguish each single shape from all the others (and what then if we decide to add new classes to be recognized?). Usually such techniques are used for experimental data which can be increased at will, and often happen to be so overwhelming in number that manual processing would be totally impossible. This may represent an issue for paleographical or other similar applications, because of the relatively low quantity of digital samples available, and poses even more problems when our aim is detecting subtle differences rather than essential patterns: for instance, if we want to train a system so that it recognizes the differences between the same letter *alpha* with or without serifs, or with

a broken rather than continuous horizontal segment, rather than training it mainly for distinguishing between different letters (an *alpha* is more obviously different from an *omikron*), this would require even more samples, and might possibly result in any case into an excessive number of system failures. So, for real-world applications of such techniques to paleographical data we must take into accounts several potential issues:

1. the very complex preprocessing required to obtain vectors describing each shape we want to use for training or recognition from a photographic image (resizing, dealing with color or grayscale information, detecting edges, removing noise, etc.). Also, the required graphical preprocessing especially for manuscripts should cope with a very 'noisy' background and with very different sizes, shapes and locations of letters. In some scenarios it might be possible to exploit existing digital data if available, but even then the image-letter(s) pairs to be used to feed the training process should meet some basic requirements (in terms of resolution, colors, shape isolation, etc.) which would probably require additional processing.

2. the scenario is even worse when applied to manuscripts rather than to epigraphical material, as for obvious reasons in the former case the shapes are usually much more complex and ligatures and abbreviations define new graphical signs which we should treat like individual letters, as they may bear no resemblance with their simple components. Thus, the target alphabet to recognize would be much bigger, including not only the simple letters but also several of their peculiar combinations; in other terms, we would have a much bigger output layer in our network.

3. also, if we want to experiment with different variants of the same shape, things get even worse as the nuances may be so subtle in the general context of a shape that this would require a very troublesome learning: probably the required samples would be far more than the samples we could provide, the processing performance would be seriously degraded and the results inadequate, as there would be too much noise activity in the output layer. The very nature of such techniques rely completely on learning by samples, and when samples belonging to different classes are too similar each other it would be impossible for the machine to grasp their differences; we would have to raise the similarity tresholds up to a point where each sample would be judged as representative of a different class. In this case the only way to go should probably be a much wiser preprocessing, even manually driven, which selects just the traits we judge as relevant for our purposes: but a similar work might easily result too expensive to be rewarding.

In a similar scenario, probably a more proficient approach should take into account a number of helper methods and start from the assumption that neural recognition is best fit to figure out and compare essential patterns from large sample data rather than distinguishing subtle nuances among similar types. In practical terms this means that first of all we would probably have to think about a number of pre-processing steps which help defining a priori which traits of each sample should be treated as more relevant, thus reducing all the noise data present in the original sample. This does not only include generic digital image processing as explained above; on a more specific ground it would be probably wise to preprocess the resulting images so that the system gets some well-directed clues about the features which scholars themselves define as relevant for the definition of a pattern.

This often sounds odd to people in connection with neural systems, but such systems especially in these contexts probably would never be proficient unless directed, and it's rather obvious to remark that any digital tool is not meant to replace scholars, but only to be useful for their research. As for any other technology, such systems are no magic and must be applied to the right problems and satisfy a number of requirements. In a paleographical scenario it would be probably necessary to isolate the traits which scholars define as essential for the classification of a sign, dropping all the other features which rather produce just noise; for instance, this might mean erasing some portions of each sample letter to keep just the traits which are essential to the definition of each sign, and then let the system train on such 'mutilated' samples rather than on the full picture. Automatic or semi-automatic specialized systems might be devised for such preprocessing, but the main point is that it is up to the scholars judgement, the kind of material and the edition purposes to define them in the form which best fits some specific requirements. In this context a neural system might anyway prove useful as it provides a mean of defining patterns in a more abstract way, without having to explicitly define a fully exhaustive list of models and devise some (maybe complex) digital format to store and compare them. To make a trivial sample, think of a number of letter shapes: if we were just interested in some very selected aspects we could even dispose of images, and rather describe them with a variable set of attributes, which may or not be present in this essential description of each letter (e.g. serifs, horizontal angular traits, shading, etc.), and still have a searchable system without even requiring neural techniques. Anyway, we might require to go further and take into account a greater number of traits for each shape, and we might even not be able to list them in advance, nor to say whether some of them can be treated as equivalent or not, just because we still do not know the whole corpus with all its thousands samples. In such cases it might be useful to provide a neural system with some well-preprocessed samples and leave it the job of building some patterns from them, thus providing some sort of fuzzy comparison less constrained to predefined sets of attributes. Of course, the main job of the scholar in this context is defining a theorical framework which provides the clues to the system about what is more or less relevant in any given type of sample data; this framework might often be just a working hypothesis to start with, as this analysis process typically implies adjusting "on the spot" a number of sensitive parameters which affect the general appreciation of observed data and repeat the process several times until

the best result is attained. After all, this is the typical way of working in most fields where information technology is applied to human sciences: in my opinion, one of the most difficult yet intriguing aspects of such applications is right the requirement of a degree of formal definition of every theorical aspect strict enough to be applicable to machine analysis. Of course often it's not possible to even attain such definitions in theory, but at any rate we must provide at least a good working hypothesis which fits well enough the purpose of our research and just "works" as we can reasonably expect in our digital product.

Theorical Frameworks and Preprocessing: a Textual Analogy

The samples of such way of working could be easily multiplied, but I think it might be useful to quote just one to stress this essential point, recurring to the realm of text, which usually is more easily understood and explained in a few words. I have already stressed the concept of a digital "edition" which can be the basis of any other specialized application, like e.g. metrical analysis of the texts collected in some corpus. A component of my expert system devoted to such problems refers to the observation of alliteration, which is much more relevant in Latin than in Greek texts, but can also be used in connection with other languages and researches[3]. Of course a full discussion of this component would be out of the scope of this paper, but here I'd just like to stress the same methodological implications already referred to the problem of applying neural recognition systems to graphical samples.

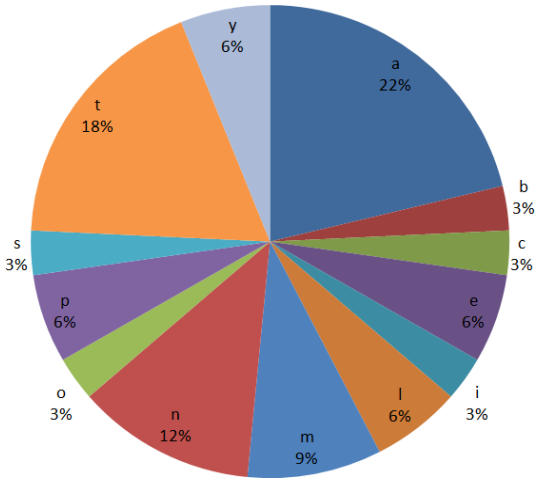
As alliteration refers to the repetition of sounds[4], a first naive approach to detect this phenomenon might just be counting all the equal sounds in a given context[5]. Even if (just for economical reasons) we don't discuss here the issues connected to defining which (even approximate) "sounds" correspond to a sequence of letters, and which of them may be treated as "equal" for the purpose of alliteration, it's obvious that such issues represent the first step in building a theorical framework for our analysis. Anyway, we must go further: the phenomenon of alliteration cannot be oversimplified to the point of just counting the equal sounds in some text. Otherwise, we might end out concluding (with Evans) that hexameter parts like *Pergama Graiis* or *talia fatur* are alliterations just because in the first sample the syllables in strong position share the "letters" *g* and *r*, and in the second they share *t* and *a*, even if their order is reversed, as we're just "counting letters". With these premises, it's hardly surprising that Evans claims that "the lines which satisfy the fundamental rule [...] are very numerous" and that for instance in Ovid's *Fasti* they are "about eighty percent of the whole" (p.44). Evans himself goes further noticing that at any rate "owing to the fact that there are only sixteen consonant or vowel sounds which cannot echo each other, it is difficult to construct a long line without a single rhyme" (p.5). In more modern terms, given that language is articulated it will be obvious that any text, whatever its extent, will be built with a very limited set of phonemes: after all, it's right the finite and very small number of phonemes which grants language its economy, and it's a trivial prediction that any text will show the repetition of these phonemes. Even to the ears of a more naive observer it should be clear enough that lines like *iam licet venias marite* (Catull. 61, 187) could hardly be said having the same acoustical effect as lines like *tympana tenta tonant palmis et cymbala / circum concava* (Lucr. 2, 618-9). This is of course the effect of the fact that the phenomenon cannot be described on a purely quantitative basis (the count of "letters"), as what is relevant is the relative position and distance of the repeated sounds and their distribution into linguistical units ("words" and syllables), rather than the (trivial) fact that sooner or later they must repeat in a text.

Should we limit ourselves to count the occurrences of each "letter" (or better "sound") in a line, we might probably grasp the existence of some alliteration when this is rather emphasized, like in the Lucretius line quoted above, but even there it appears that several 'letters' counts just add noise to our picture rather than clearly presenting it, as we can see from this chart representing the raw letters counts (as percents) in this line:

[3] The component itself has no dependency on a specific language (nor digital format), even if of course the theory behind it fits some well-defined principles; for instance, the component has been used in connection with Latin, Greek and Italian, for both metrical and non-metrical texts.

[4] Even if a true theory of alliteration seems missing from ancient authors, there are some relevant traces of their appreciation of the phenomenon; one of them is the synthetic yet significant definition in *rhet. ad Herennium* 4,2,18 "*nimia assiduitas eiusdem litterae*").

[5] That this approach has not been judged so naive as it might sound is shown by the fact that scholars like Evans (1921) did really base onto it a full 'theory' of alliteration. In his view alliteration is just a subset of what he calls "rhyme", defined as (p. xiii) "an agreement in sound between two or more syllables" which "may extend either to one letter or more", so that even words like *like* and *roll* would "rhyme" because of the shared *l*.



As you can easily see, the /t/ in this chart does not "stand out" as it clearly does when we hear this line (even if uttered with sounds which do not correspond to the ancient ones): this happens because the chart assigns the same value to each letter, rather than reflecting the hierarchy of the linguistical elements involved, their relations and the literary tradition at the base of the texts examined. Of all the letters represented in this chart only very few are really relevant for our purpose, not because of their intrinsic phonetic value but just because of their position in the text examined. Thus, most of the slices in this chart are just noise, which distract our attention from the phenomenon we have selected to analyze. If we want to provide some 'score' of the level of alliteration detectable in a text, we must first provide at least as a working hypothesis a theorical definition of the phenomenon, thus ruling out all the irrelevant 'letters', just like in ANN each neuron can 'weight' its inputs to produce the expected output. To this end, we could broadly state some chief starting points[6]:

• **phonemic analysis**: alliteration is the repetition of sounds, and as such any text requires to be analyzed in phonemic terms. Of course, this implies an approximate reconstruction of phonemic values for the texts being examined, in this sample Greek and Latin texts, whose limits in terms of chronological and geographical extent are well known, but this is a necessary limitation. This also enforces the assumption that phonetic variants of the same phoneme are treated as alliterating: thus the ground for this analysis is defined in phonological (rather than phonetic) terms. Also, it must be remembered that the Greek and Latin texts analyzed were performed for an audience rather than silently read, so that we must resist the temptation of analyzing this phonetic phenomenon as a printed text, referring to the eye rather than to the ear.

• **alliteration and language** constraints: *a priori* it is very difficult to estimate language constraints on alliteration, but in this working model I stress the (primarily word-) **beginning** sounds rather than the ending sounds, for both practical and theorical reasons. First, cases of *homeoteleuton* (which is often also an *homoioptoton*) are very frequent among connected words in inflectional languages like Latin or Greek (and in general in indoeuropean languages) where the grammatical role of suffixes is generally much wider than that of prefixes: the frequence of such "grammatical rhymes" was already noticed and discarded as irrelevant since Evans[7]. Second, the alliteration of word initial sounds is well-attested as a traditional feature which might well go back to what comparatists call the Indoeuropean metrics (it is the main metrical feature of systems like the ancient German poetry[8], and its importance in the archaic Latin poetry is well-known). Also, some morphosyntactic diachronic phenomena clearly show the importance of word-initial sounds in connecting words into combinations which later can be frozen (e.g. Italian *domineddio / domeneddio* from *domine Deus*, or the strong connection between ἦμος and the expression of solar chronology in Homer and Hesiod, enforced by the combination with ἠώς and ἠέλιος[9]).

Of course, for the general "sound" of the whole verse or sentence also inner- and final (even if much more 'forced' by the language structure) repetitions count[10], nor discussing the 'intentions' of the poet is relevant here[11], as I am looking

[6] For the alliteration proper these points mainly reflect to some extent Valesio 1967; some good examples (collected with a fine stylistic sense but in the context of a much weaker theory) are also found in Herescu 1960, who anyway devotes only a section to the alliteration.

[7] "Such rhymes in Latin are merely accidental" (Evans 1921 p.2, quoting Hor. *ars* 100-101 ending with two imperatives: ... *sunto* / ... *agunto*).

[8] Cfr. Valesio 1967 pp.25 sqq.

[9] Cfr. Valesio 1967 pp.179-181 who refers to Monteil 1963.

[10] Valesio, who too restricts (for the same practical reason of economy) his definition of alliteration, really takes into account only sequences of initial or final sounds, regarding every combination of them (i.e. all the sounds must be initial or all the sounds must be final: no combination is taken into account) or any internal repetition at most as "para-allitterazioni". He instead widens his definition when dealing with vowels, thus being forced to abandon the consideration of their relative position, probably also because of the English texts he uses as samples (for some considerations about vowels in Latin poetry see Herescu 1960 pp.84 sqq., whose approach anyway seems somewhat questionable because of the more limited role of vocalic phonemes in the Latin language, but at least takes into account the necessary distinction between 'strong' and 'weak' syllables – p.26).

[11] On this questionable aspect cfr. e.g. Herescu 1960 p.128: "même si les rencontres phoniques se produsaient d'elles-mêmes, à l'insu de l'écrivain, il reste néanmoins que celui-ci part avec, dans l'oreille, un certain enchaînement de sons, une certaine musique, qui se réalisera dans ses vers même sans qu'il s'en rende compte. Les sons, dira-t-on, ont leurs propres intuitions et le poète est un «peintre aveugle» (Michel-Ange)".

for an essential pattern detection function. Also, I prefer to devote to the tradition of the rhyme and its ancestors a different set of analysis functions, which for Classical times play little or no role in Latin and Greek poetry and anyway mostly relates to another context (typically alliteration is an intrastichic and rhyme an extrastichic feature). Thus I mainly define (detect) alliteration in terms of beginning sounds, looking to the other repetitions only at a second stage, and walking from "words" down to syllables and phonemes. The main pillars for building up an alliterating sequence are thus considered the "words"; inner (syllabic and phonemic) alliterations are taken into account only in dependency of word alliterations. Finally, more generic euphonic considerations are not of concern when dealing with this strict working definition of alliteration: thus I exclude the analysis of repetitions of vowels whatever their position may be, unless they are involved in the alliteration as already defined (e.g. words beginning with the same vowel, syllables beginning with the same vowel, vowels echoing the main alliterating vowel).

• the resulting **hierarchy** of linguistical units for this analysis is thus: sentence, word, syllable, phoneme, and it is walked from top to bottom. The sentence (or line in metrical texts) defines the analysis context; words beginning with the same sound(s) define alliterating sequences; in each of these sequences I extend the analysis of repeated initial sounds to each syllable of each word, and finally add some weight to the "echoes" of the main sound in the remaining (i.e. not included in initial repetitions) phonemes of each syllable.

• the **context** for alliteration is defined as the single verse in metrical texts and the single sentence in non metrical texts. This does not rule out the fact that sometimes alliterations can be carried over more lines, but this is a secondary feature when dealing with an essential detection function, and belongs to a wider analysis context which is not my primary concern at this stage.

• **leftwards scan**: each word in a sequence is compared with any of the words at its left in the same sequence, taking into adequate account the number of the initial segments which are considered as equal for the purpose of alliteration and their relative distance. This is compliant with the assumption that most of the ancient poetry is defined in acoustic rather than in visual terms, as it is orally performed for an audience: thus the repetition of sounds becomes apparent once the same sound pattern is repeated by newly uttered words: the word just uttered can be traced back to all the words already uttered in the same portion of the speech (typically a verse or a sentence), but it cannot be compared to what it is still to come in the speech. In other words, the ear can go leftwards, not rightwards as the eye.

• as for detection of the alliteration I make no attempt to distinguish between **iconic** and non-iconic alliteration, which relates to style and may involve a certain level of arbitrariness[12] (and for the same reason is also difficult to be understood by machine analysis).

Without delving into details, the algorithm I devised from the theorical framework sketched above acts on text portions defined on linguistical and metrical grounds, the sentence (in non-metrical texts) and the line (in metrical texts), which thus define its context of application. For instance, we will analyze a single line at a time, like the hexameter quoted above, *tympana tenta tonant palmis et cymbala circum*. According to the principles stated above, I start detecting alliterating sequences, defined as all the (not necessarily adjacent) "words" beginning with the same sound(s), which in this sample are *tympana tenta tonant* and *cymbala circum*. This approximation reflects the paramount role of initial sounds at the beginning of words as the building pillars of an alliterating sequence; it must be noticed that not all the words need to be adjacent each other, but typically at least a predefined number of them do (as we want to detect the repetition of initial sound(s) even when there is some other word between two words in the same context, but we don't want to treat as an alliterating sequence a sentence where e.g. the first and last word separated by several words happen to begin with the same sound): this number is a parameter for the scorer (a "sequence treshold"), typically set to 2; clients can customize it to change the scorer sensitivity to the accumulation of adjacent words. For each sequence I compare the initial sound(s) of each word to the initial sound(s) of each word to its left, picking up the pair with the longest portion in common and recording its length (the more sounds in common, the higher the effect) and relative distance (the nearer the words, the higher the effect).

This procedure takes into account the initial sounds of each word in the sequence as the basis for the repetition, according to the relevance of word-beginning sounds defined in the above theory and to the assumption that most of the ancient poetry is defined in acoustic rather than in visual terms, as it is orally performed for an audience: thus the repetition of sounds becomes apparent once the same sound pattern is repeated by newly uttered words: the word just uttered can be traced back to all the words already uttered in the same portion of the speech (typically a verse or a sentence), but it cannot be compared to what it is still to come in the speech. I then apply at the level of the syllable in the context of a word the same procedure already applied at the level of the word in the context of a sentence (or verse), walking down the linguistic hierarchy: each syllable is compared with all the left syllables in the same word, but also with the first syllable of the first word of the sequence. This produces an array of counts of equal initial segments for each syllable, together with their relative distance, and constitutes what we can call the "syllabic heads" of a sequence. Finally, the segments of each syllables not included in the previous analysis (the syllabic heads) are also examined to see whether any of them is equal to the 'master' segment in each sequence, which is simply the first segment of the first word in the sequence itself. For instance, in the sequence *tympana tenta tonant* the word *to.nant* has the second syllable containing the same /t/ segment beginning the sequence. The occurrence of this sound, even if not initial (which is

[12] For some examples see Herescu 1960 pp. 108 sqq.; for the discussion about *onomatopoeia* see also p.125 n.2.

already taken into account by syllabic heads), has somewhat the effect of 'echoing' the dominant sound in the sequence and thus enforces its perception. This constitutes what we can call the "segmental echoes" of a sequence.

At this stage I have considered all the linguistical levels from "word" to syllable and phoneme, taking into account their position and distance, with a set of procedures which reflect the theorical framework sketched above; once this analysis has been completed, I take into account the word heads, syllabic heads and segmental echoes of every sequence in an input text (either a sentence or a line) combining them into a single numeric score, using a set of parameters which assign a specific weight to several aspects of the algorithm, like word and syllabic heads distance and phonemic echoes; the greater the score, the higher the alliterating effect. Clients can change these parameters to variously adjust the scorer sensitivity to the texts being examined and the purposes of the analysis: typically scorer results on sample texts can be used at this stage for fine-tuning these parameters and then repeat the analysis several times until the best result is achieved.

To sum up, this sample starts from what might be considered a simple problem (detecting some repetition of sounds) to show the consequences of a trivial analysis ("counting letters"), which simply tries to swallow all the available data with no preliminary evaluation of their relevance on theorical grounds: a casual analyst might well be tented by the raw computing power at his disposal to just count all the letters in every sentence of his texts to detect some alliterations in them. Of course, a more sophisticated analysis would refer to sounds rather than to letters, for the trivial fact that for historical and practical reasons no written language faithfully represents the corresponding sounds with a simple one-to-one ratio between letters and phonemes. In this case, a possibly complex preprocessing would be required to output e.g. a sequence of IPA characters from a sequence of a national alphabet letters. Even then (counting IPA characters rather than generic letters) the appreciation of the alliteration would probably be too inaccurate, as the analyst would just be facing a number of raw counts among which it would be very difficult to grasp what he can easily perceive by his ears: this is the situation already depicted by the pie chart sampled above, where each sound is given the same weight, while it should be clear that its position is the first factor defining the phenomenon under analysis. With such treatment probably the most emphasized alliterations might be detected, but we would also get a noise level so high that we might often be misleaded in not only underestimating but (probably worse) overestimating several cases (remember the Evan's samples like *Pergama Grais*).

In other words, not all the sounds have the equal weight for defining an alliteration, so several of them just represent noise data in a text: excluding them or at least reducing their weight it is not some sort of 'trick' or adjustment we should care of before handling data to a machine, just because otherwise it would not 'work' as expected; excluding them is right a consequence of a well-defined theorical background, according to which it would be an error to do otherwise, an error which would *a priori* invalidate any experimental result, whatever it might be. We have seen that alliteration is a complex phenomenon and that in order to give at least a working definition of it we must recur to linguistics and metrics, taking into account sentences, words, syllables, phonemes (rather than allophones), their 'equality', positions and distances; and we cannot expect from a machine to 'understand' all these aspects with no instruction, just feeding it with a bunch of raw characters. It's up to our theory to define the relevance of each of these characters in the big picture we're trying to sketch, assigning them a well-rounded weight. Of course, probably it will be difficult to do this since the very beginning of our analysis: as a matter of fact, the results themselves will help us in fine-tuning all the parameters involved in our detection algorithm so that it 'works' best for our texts and our purposes: we might want to give more or less relevance to the number of words which should be adjacent to define an alliterating sequence, to the count of phonemes words or syllables share, to their relative distance, etc., so that our analysis is more or less sensitive to certain types of alliterations.

The same applies to the usage of neural recognition systems in contexts like paleography: preprocessing here is not just a remedy for an otherwise poorly performing system. As we cannot expect no magic in detecting alliterations if we just feed a system with a bunch of letters, we can no more claim that a neural recognition system might be able to perform well with a relatively small sample where each shape is fed as it is, without instructing the system about the features scholars define as relevant for their data and purposes. As for the pie chart sampled above we might well get some positive results even in the worst context, but the noise level and consequently the number of failures would be so high that such a system would be as unpractical for our purposes, just like the above chart for detecting alliterations.

It must be also emphasized that the directions given to the system via preprocessing would probably be different according to the data being analyzed and to our purposes in analyzing them. The alliteration sample shows that different languages and literary traditions might well require a different adjustment of the various parameters, or even more significant changes in the algorithm itself: if we just refer to the theorical points sketched above we can see that several aspects directly derive from various linguistical, metrical and literary assumptions (e.g. the role of suffixation in indoeuropean morphology and its consequences for omeoteleuton and rhyme; the role of consonants vs. vowels in alliteration; the role of words, syllables and phonemes in alliterating sequences; the relevance of word-initial sounds in poetic traditions like old German and ancient Latin; the importance of recitation in ancient poetry; etc.), which might have different weight in different languages, texts, literary genres, etc. Also, such software systems are created with specific purposes, and often it is important to adjust their sensitivity according to them: for instance, running an alliteration detection system on a literary Italian corpus of prose texts where for some reason we might want to look for alliterating sentences would require a higher sensitivity than running it on a corpus of early Latin poetry.

The same applies to paleographical applications, where our factors would rather be the types of shapes involved and their level of complexity and similarity, but also our purposes in setting up a recognition system: a system designed to recognize just different letters might probably require to be less sensitive than one designed to detect relevant differences between shapes of the same letter. If our data are not enough to provide a good number of samples we might even want to define some automatic distortion procedures which artificially increase them by applying to each letter shape some slight distortions which do not affect their relevant features. Even then, we would always refer to our theorical framework, which would be the only judge for defining which features can be treated as relevant (and thus cannot be touched by the distortion process) and which instead can be freely altered, just like for alliteration our principles define which letters are relevant, and to what extent, and which represent just noise.

Digital Editions: Possible Approaches

With these premises, it seems very likely that such neural systems might find different types of application and success in different scenarios, and at any rate it's not possible to predict their outcome until they are tested "on the spot", fine-tuning their parameters and eventually even the theorical framework behind data preprocessing until we get the best results, according to the data type and the system purposes. As for any other technology, there is nothing magic here, and such systems are orders of magnitude far behind the capacity of judgment of human beings: after all, we are not supposed to, nor we would probably want to, find some way of avoiding to think and set up a theorical framework by ourselves. Such tools are just a convenient way of dealing with very large amount of data, and it's only up to scholars using them with judgement and profit. To sum up, the problem with paleographical scenarios is that usually the most suggestive applications would require far too many samples to be practical or even possible, given our relatively limited set of data and/or the effort spent in their preprocessing: if we want our machine to recognize different letters like an OCR software we would have to lower our 'similarity' treshold, so that the variants can be easily recognized as such; but if at the same time we also want to recognize chronological, regional or personal variants of each different shape we would have to raise the treshold, may be up to a level where the machine would be so picky that each shape would be judged as a different letter; finding the right balance would be very difficult if not impossible.

Rather, in a practical implementation we might think of partitioning the problems and limit the application of such techniques to patterns subsets, referring to a carefully defined theorical background, just like in the alliteration sample: for instance, we might think of applications where visuals are even more compelling, like cataloguing brick stamps. In this case we're dealing with letters of abbreviations which often are so connected and reshaped into monograms-like shapes that they can be treated as single drawings; of course even with stamps it might often be the case that differences are too fine to be efficiently detected, but given their high number it could be useful to have a system trained just to recognize some single traits of each of the main classes, maybe just to provide users of a digital catalogue a quick and approximative way of detecting the major classification of a sample. In this context not only an algorithmic approach describing each single shape would be extremely impractical, but even providing some sort of input for a hypothetical query system would be difficult: for instance, once we have defined a subset of relevant shape traits extracting them from the much more complex shapes of the actual samples we might think of presenting a relatively long list of them to the user to pick from; but this would probably be a cumbersome way of querying our catalogue, as users would have to carefully browse our samples list until they find what they are looking for. Even if the count of such more generalized samples were not so big, users would require a certain effort only to make a simple query.

A more user-friendly approach instead might be providing users with a virtual board where they sketch with their mouse (or other pointing device) just the traits we have defined as relevant for our super-classes, rather than looking for them in a relatively long list of prebuilt samples: then a neural network trained to recognize just these traits might be used to select the class of the sample being queried. This is a purely hypothetical sample, but it shows how an ANN might even be used to just improve the users experience in defining their input for querying a digital catalogue: here the classes to be recognized would be much less than the actual stamps patterns, because editors would have restricted them to a much more abstract subset, defining only their most relevant traits; but this might be enough to restrict the query to a more significant set of classes (which might be later again restricted by means of similar or different query parameters, probably a combination of visual and non-visual ones), and this would be accomplished with a user just drawing something, and letting the ANN recognize the most similar pattern in its trained set.

Here again it would be up to the scholars' publishing the catalogue to define which traits should be selected as the most relevant partitioning criteria for their data, according to the theorical framework they have set for their research method. Going one step further, this "partitioned" approach might also prove useful in paleographical scenarios, where at a first approach it is not so easy to think of "decomposing" letter shapes into isolated traits which might not even make any sense for a human reader. Here a broader analogy might help: think of a typical identikit procedure, where witnesses must help building up the face of a person by selecting each trait from a set of galleries: they select head shape, nose, lips, ears, hairs etc. one at a time from separated galleries of samples, picking one from each of them and then assembling all the components into a hypothetical portrait. As for letters, we have seen that in several cases we might have to face issues originated by the limited set of available samples in contrast with a high number of classes to be recognized, often distinguished by nuances which are too fine to be efficiently manipulated. Anyway, we might think of partitioning the problem in advance, even before letting an ANN enter the scene, and define a set of single features we consider as relevant for distinguishing at least the most important classes: for instance we might extract from letters

just the shape of their serifs, the orientation of some selected segments, the shape of elliptical parts, etc., and define each as a separate step, being treated by a specifically trained ANN. Like when building an indentikit, users might just literally draw a sample for each of these features and let several trained ANN's recognize their patterns. Each of these steps would be a progress towards the final result, which would thus lead to some sort of classification of the letter sample. This way the letter would no more be treated as a single pattern, but rather decomposed into a set of separated patterns, each recognized by a differently trained ANN. Later we might be able to sum up the results of the output of all the ANN's, eventually combine them with other (non visual) query parameters and get the desired classification. Of course this is just a hypothetical scenario, which would probably be practical only when dealing with a large amount of data and classes, or when there would be no feasible alternatives to a purely visual way of querying a database of patterns; but it might be more rewarding than letting the machine try to perform a good recognition on data which are intrinsecally very difficult to be treated for training.

Daniele Fusi
University of Rome "La Sapienza"

Bibliography

Chang, Chih-Chung and Lin, Chih-Jen. "LIBSVM -- A Library for Support Vector Machines." <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

Evans, W.J. Alliteratio Latina or Alliteration in Latin Verse Reduced to Rule with Special Reference to Catullus, Horace, Juvenal, Lucan, Lucretius, Martial, Ovid, Persius, Phaedrus, Priapeia, Propertius, Statius, Tibullus, and Virgil. London, 1921.

Fusi, D. "Edizione epigrafica digitale di testi greci e latini: dal testo marcato alla banca dati." Digital Philology and Medieval Texts. Ed. A. Ciula, F. Stella. Pisa, 2007. 121-163. <http://www.fusisoft.it/Doc/ActaArezzo.pdf>

Fusi, D. "An Expert System for the Classical Languages: Metrical Analysis Components." (to be printed in the Proceedings of the international conference Trends in Computational and Formal Philology - An Italian Overview. Venice and Padua, 22-24 May 2008.) <http://www.fusisoft.it/Doc/ActaVenezia.pdf>

Gersherson, C. "Artificial Neural Networks for Beginners." <http://arxiv.org/abs/cs.NE/0308031>

Herescu, N.I. La poésie latine. Étude des structure phoniques. Paris, 1960.

Monteil, P. La phrase relative en grec ancien. Paris, 1963.

Stergiou, Chr., and Siganos, D. "Neural Networks." <http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html>

Valesio, P. Strutture dell'allitterazione. Grammatica, retorica e folklore verbale. Bologna, 1967.