An Experimentation Line for Underlying Graphemic Properties
Acquiring Knowledge from Text Data with Self Organizing Maps

Gilles Bernard, Nourredine Aliane and Otman Manad
Laboratoire d'Informatique Avancée de Saint-Denis (LIASD), PARIS 8 University, 2 Rue de la Liberté, Saint-Denis, France

Keywords: Neural Networks, Computational Linguistics, Unsupervised Knowledge Learning, Text Information Retrieval.

Abstract: We present an experimentation line that encompasses various stages for research on graphemes distribution and unsupervised classification. We aim to help close the gap between recent research results showing the abilities of unsupervised learning and clustering algorithms to detect underlying properties of phonemes and the present possibilities of Unicode textual representation. Our procedures need to ensure repeatability and guarantee that no information is implicitely present in the preprocessing of data. Our approach is able to categorize potential graphemes correctly, thus showing that not only phonemic properties are indeed present in textual data, but that they can be automatically retrieved from raw-unicode text data and translated into phonemic representations. By the way, we observe that SOM algorithm copes well with very sparse vectors.

1 INTRODUCTION

This paper presents an attempt at an experimentation line that addresses the issue of discovering oral and scriptural properties of graphemes without any prior knowledge, with the support of big corpora, within a neural net framework, namely Self Organizing Map (Kohonen, 1995) among other tools.

Graphemes have properties which partly reflect the properties of their underlying oral correspondents (phonemes), and partly scriptural formatting uses and conventions, as well as morphemic properties. But graphemes, as well as they are known to human beings, are not immediately accessible to computers; the only unit the computer processes is the computer character – not even an equivalent to human written character.

Research has been done, essentially by computational linguists, that address issues related to our problem, with proposals of algorithms effectively tested against linguistic data, beginning with the seminal work of (Harris, 1968). Of particular interest here is a paper by (Goldsmith and Xanthos, 2009) that has shown the feasibility of automatic detection of some phonological phenomena in any language, given a corpus in phonemic representation.

On the other hand, the devising of Unicode conventions has given computer scientists the ability to work on a close approximation to written characters in most languages. But there is still work to be done to reduce the gap between human produced phonemic representation(s) and the Unicode representation(s). Our results show that experiments on the structural and distributional properties of characters may give us means to access the properties of graphemes and their underlying phonemes, thus contributing to this gap reduction.

This approach would enable us to generalize and validate approaches as the one quoted above. Among unsupervised algorithms, Self Organizing Map algorithm has been selected for its ability to map in two dimensions the results, with easier detection of the phenomena analyzed.

2 THE ISSUE

Among digrams 'ca' is more frequent than 'tc' because of the properties of consonants and vowels; but 'ch' is more frequent than 'ct' for a completely different reason: 'ch' is in fact one consonant written with two characters. The properties of 'ch' (its distribution) are thus to be compared with those of 'c' and not with those of 'ca'. Other properties relate to scriptural (as opposed to oral) properties, as uppercase vs lowercase or punctuation.

Usual entropy measures or Markov chains do not capture such differences, which is not such a drawback from an empirical point of view, in helping with strings pronunciation, speech recognition, error cor-

Bernard G., Aliane N. and Manad O..
An Experimentation Line for Underlying Graphemic Properties - Acquiring Knowledge from Text Data with Self Organizing Maps.
DOI: 10.5220/0005577706590666
In *Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics* (ANNIIP-2015), pages 659-666
ISBN: 978-989-758-122-9

rection, language identification, or optical recognition. But it constitutes a hindrance for linguistic or psychological research and in deciphering unknown languages or codes. In such cases we need more refined analyses.

Graphemes (written basic units as 'c' or 'ch') seem to be processed as units by the typer as well as by the reader, as shown by psychological studies such as (Weingarten et al., 2004). Interestingly, for (Jones and Mewhort, 2004), the difference between lowercase and uppercase characters frequency induces different psychological properties: "[...] identification time for uppercase letters is predicted better by frequency of uppercase letters than by frequency of lowercase letters".

We choose to take nothing for granted, not even separators; as the 'h' of 'ch' a white space could well, in some writing systems, be part of a complex symbol. In English a dot '.' finishes a sentence or signals an abbreviation; it can also be part of a grapheme, as in '...', in which it does not retain its usual function.

The aim of our experimentation line is to capture those properties without making any educated guess (as what is uppercase or what is a separator), and see how much information can be gathered.

3 STATE OF THE ART

Interest in the distributional properties of written data is not knew. The famous Zipf law on the statistical properties of words was discovered as early as 1912 by Estoup and given a mathematical formula by Zipf (Petruszewycz, 1973). Shannon opened new areas with information theory, and in the 1960's, linguists and mathematicians have worked on related issues: mathematical generalization of Zipf law (Mandelbrot, 1957), vowel / consonant automatic distinction (Sukhotin, 1962), prediction of morpheme closure (Harris, 1968).

Those issues underwent a loss of interest in favour of syntactic and semantic parsing. Democratization of computers renewed this interest in the 1990's, with a variety of statistical approach to low level phenomena, see (Powers, 1997).

But in their vast majority, researchers in empirical work on raw data have more concentrated on the properties of words, than on grammatical data or graphemes (with the exception of handwritten grapheme recognition, which has very little to do with our issue). Grammatical data was abstract and obscured by years of so-called deep structure (but see (Bernard, 1997) using statistics of grammatical data to capture word properties); graphemes were too trivial. Furthermore, works like Harris' algorithm for detecting meaningful units based on their distribution waited for years before being rediscovered (Bernard and Mariage, 2005), (Pham and Lee, 2015). The same can be said of Sukhotin's algorithm, known only in a very narrow field (Guy, 1991).

Recent research has seen new approaches. (Goldsmith and Xanthos, 2009) have successfully assigned, from a corpus in phonemic representation, a phonological category for each phoneme, as vowel vs consonant or low vs high vowel (in vowel harmonic languages, see also (Goldsmith and Riggle, 2012)), using distributional profiles, Hidden Markov Models or Spectral Clustering. (Mayer and Rohrdantz, 2013) also show the possibilities of another clustering algorithm (Ward, 1967) for discovering sound properties, but, as with the others, "the words should be given in some phonemic transcription (e.g., using IPA)".

This constraint on phonemic representation is what blocks the possibility of adapting such algorithms outside of linguistic research, for computing on raw data, because computer applications do no use IPA (International Phonetic Alphabet) but Unicode (at best).

4 SYSTEM DESCRIPTION

4.1 Graphemix Overview

The experiments presented in this paper have been produced by Graphemix 0.1 (figure 1). Our system aims to encapsulate every stage of experiment production, from the constitution of the corpus to the visualization of the results, going through parameter selection for algorithms and monitoring of the database.

The components are written in C++ using Qt 5 libraries; it is available as a gitlab project (https://gitlab.com/harris/charprops, with cmake build system), running on *nix systems.

It presents itself as a board with four tabs: corpus constitution, corpus analysis, algorithm choice and tuning, result visualization. In the first tab, the user composes a corpus from the web and local files, with format and encoding preferences. The second tab contains the customizable analysis of the corpus into potential graphemes, as well as primary statistical results. In the third tab one selects an algorithm and tunes its parameters. Version 0.1 only has Sukhotin's algorithm and Kohonen's Self Organizing Maps. Eventually the fourth tab yields the results. We describe below with more detail the components relating to each tab.

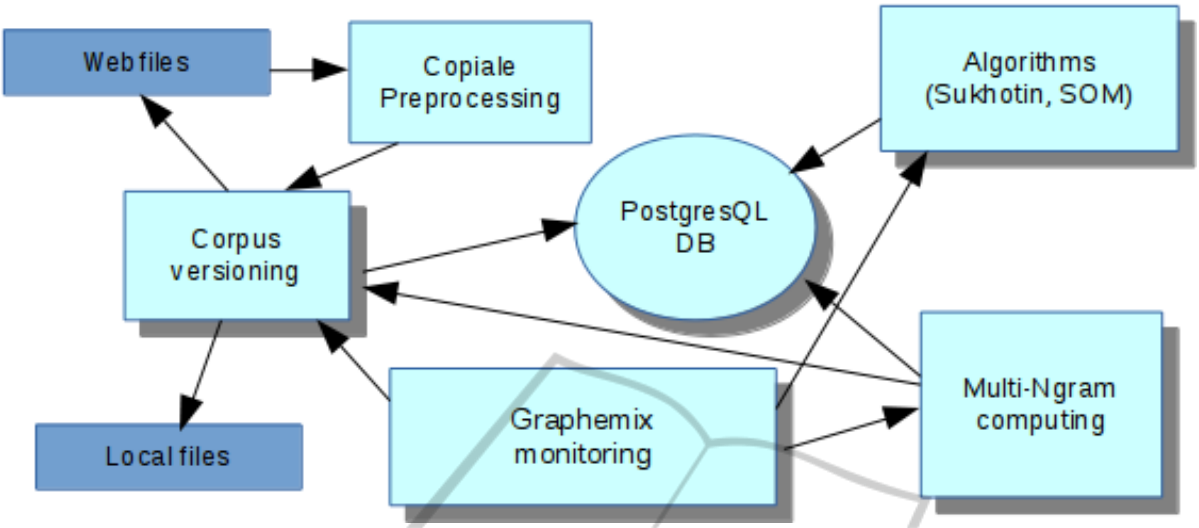

Figure 1: Experimentation line.

4.2 Corpus Constitution

Corpus constitution is done with a corpus versioning tool inspired by the operation of git. All data are stored in a PostgreSQL database.

Though working on Unicode corpora is our goal, in Graphemics 0.1, only written languages in the Unicode Basic Multilingual Plane are supported, either in UTF-8, UTF-16 or CS-4 representations. The user can select other charsets, or introduce arbitrary Unicode representations for codices that are not accepted by the Unicode Consortium.

4.3 Corpus Analysis

The analysis takes three steps: at first, extracting every possible grapheme, let us call it a candidate, then compute what are the candidates that occur after one candidate and their number of occurrences in this context.

The result is a two dimensional matrix, whose lines are the candidates and the columns their neighbour candidates, and the values are the number of times each candidate occurs after another. Transposition of the matrix gives the number of times each candidate occurs before another.

This matrix is the input for the algorithms in the following tab. The first two steps are done together, for performance reason; with big corpora, it would not be reasonable to separate identification of candidates and identification of their context. The next subsection details those two first steps and the following one the produced matrix.

4.4 Look up Candidates

We divide the texts in candidates, that is, in characters ngrams, whose size is comprised between 1 and N. The N upper bound can be selected by the user; it is set by default to 5.

If two letters graphemes (English ‘sh’, French ‘ch’, German ‘ie’) are frequent, three and four letter ones are not unusual: English ‘ough’, French ‘eaux’, German ‘sch’, punctuation ‘...’. The default value of five seems to be a good general upper limit.

We determine all possible graphemes with their right context. For example, the sequence:

seeing through

whose grapheme analysis should yield:

‘s’ ‘ee’ ‘i’ ‘ng’ ‘ ’ ‘th’ ‘r’ ‘ough’.

We first posit a window of ten characters (the double of the upper bound size):

seeing thr

Then we take every candidate on the left and every candidate as a possible neighbour on the right. We take the size one candidate with all possible size neighbours on the right. We then do the same with the size two candidate, and so on until size 5. The last possible analysis is “seein+g thr”. Table 1 illustrates this (‘_’ replaces space for legibility).

We then move the window one character to the right and recurse. With our corpus of over five millions of characters, we had 140 billions of possible contexts. Most of those possibilities include the same candidates, in candidate position or in neighbour position, so we must take into account the identification

Table 1: Analysis of a sequence.

Candidate	Neighbour
s	e
	ee
	eei
	eein
	eeing
se	e
	ei
	...
	eing
see	i
	...
	ing_t
...	...
seein	g
	...
	g_thr

time for counting cooccurrences of the same candidate with the same neighbour.

In order to reduce the computation time and memory usage, we have used a character prefix trie representing the ngrams and their contexts. The depth of the trie is at most 10, and its breadth is at most the number of possible Unicode characters in the BMP (theoretically 65,536).

4.5 Build the Matrix

The context matrix is a sparse matrix with immense size, as candidates could theoretically be numbered in millions. Fortunately, not every possible ngram occurs in the corpus; and lots of ngrams only occur a few times.

Ngrams that occur few times in big character corpus are not likely candidates for being graphemes, so they can (and should) be eliminated. But we made the choice not to apply any apriori threshold, so even hapax (occurring only once) ngrams are taken into account in the matrix. It is only in a following stage that the user selects the threshold to be used for SOM or Sukhotin usage.

The afore mentioned PostgreSQL database stores the components of the matrix (as {i, j, value} records), as well as other values (e.g. threshold). All requests from the algorithms are made to this base.

4.6 Algorithm Panel

The algorithm tab is actually very simple, as only two algorithms are available in version 0.1: Sukhotin's algorithm and Kohonen's Self Organizing Maps. We will only dwell on the second, as the first one cannot be used (for separating vowels and consonants) before identification of the graphemes.

4.6.1 Self Organizing Maps

Within the unsupervised neural models, Self Organizing Maps (SOM) is probably the most popular one nowadays, so there is no need to give a detailed description. See (Kohonen, 1995) for a complete description of the algorithm.

The SOM learning algorithm extracts the main characteristics of the data space and makes a projection of those properties from a high-dimensional space onto a bi-dimensional map. The most interesting property of this mapping is that it preserves the topological ordering of the implicit relations between the data categories. It depicts similarity between data classes by encoding them in the proximity between clusters of units distributed over the map: related forms in the data space are near one another in the map space.

The dimensionality reduction of the extracted representation allows to discover and analyze relationships between data classes that would be impossible to detect in their original space. Moreover, the bi-dimensional mapping provides a convenient visualization interface.

X being the input vector, j an index on the neurons, W the memory vector of the neurons, the winner, J*, is determined by the following equation, where d() is a distance or similarity measure:

$$d(X, W_{j^*}) = \min_{\text{for each j in } \{n\ ;\ p\}} d(X, W_j) \quad (1)$$

The distance can be euclidian, Manhattan, or some other. The cosine similarity (dot product, equation 2) is often used; with sparse vectors it can drastically reduce computation time.

$$C = \sum_{i=0}^{n-1} X_i W_i \quad (2)$$

In the learning phase the winner and every neuron in its neighbourhood are modified according to equation 3, where t is the time (epoch), j an index on the neurons:

$$W_j^{(t+1)} = W_j^{(t)} + \alpha^{(t)} V_r^{(t)}(j, j^*)(X_i^{(t)} - W_j^{(t)}) \quad (3)$$

The learning rate α follows the equation 4, where α_0 is the initial learning rate.

$$\alpha^{(t)} = \alpha_0 (1 - \frac{t}{t_{max}}) \quad (4)$$

Learning in the neighbourhood V of the winner follows the gaussian variant given by equation 5, which yields better results than mexican hat or other variants.

$$V_r^{(t)}(j-j^*) = e^{-\frac{dm(j,j^*)}{2\sigma^2(t)}} \quad (5)$$

dm: distance Manhattan

with σ obeying equation 6, where i = initial value and f = final value.

$$\sigma^{(t)} = \sigma_i(\frac{\sigma_i}{\sigma_f})^{\frac{t}{tmax}} \quad (6)$$

Uses of SOM applied to very high dimensional data spaces (we have experimented here up to half a million dimensions), as well as applications on symbolic data, are not so frequent (but see (Mayer et al., 2008) and (Tsimboukakis and Tambouratzis, 2007)). In previous studies the dimensionality was reduced by various ad-hoc means (e.g. (Honkela et al., 1997), (Bernard, 1997)).

This well known "curse of dimensionality", see for instance (Aggarwal et al., 2001), could fully apply here, as we can easily have millions of dimensions, with very sparse vectors (in our runs, 1 non zero component in 5,000). In such cases the authors recommend lower norms (L1 or even fractional norms). If the same reasoning applies here, Manhattan could be a good choice here; so we added it to the possible choices: euclidian distance, manhattan distance and cosine similarity.

To assess the quality of the classification, we compute the intra-class inertia which measures the homogeneity inside groups, hereafter HI (equation 7), and the inter-class inertia, which measures the heterogeneity between groups, hereafter HB (equation 8). S is the similarity measure chosen (distance or cosine similarity), m the number of groups and n_j the cardinal of each group.

$$HI = \frac{1}{m}\sum_{j=1}^{m}(\frac{1}{n_j}\sum_{i=1}^{n_j}S(X_i,W_j)) \quad (7)$$

$$HB = \frac{1}{m^2-m}\sum_{j=1}^{m}\sum_{i=1}^{m}S(W_i,W_j) \quad (8)$$

With distance measures, HI should be closer to 0 and HB closer to 1; the reverse holds with similarity measures.

5 EXPERIMENTS AND RESULTS

5.1 First Results

We have processed a French corpus composed of nine novels (5,2 Mb) extracted from Gutenberg Project; a subset of 2Mb size was used for our first tests. Choosing corpora in a well known language should make it easier to refine the processing protocol. The size of our corpus may seem little for the corpus processing community. But consider that the unit studied is the grapheme, not the word nor the phrase, and that we extract every possible candidate and all possible neighbouring candidates.

Text ending excluded, every letter has five candidates possible to its right (same to its left). It can be the last letter of five possible candidates. All these potential relationships are computed and stored.

Generation of the prefix trie containing all possible vicinities does not take much time (between five and ten minutes on an ordinary medium level computer). The toll is on the identification of which candidate is neighbour to which, and on the transactions (even in batch mode) with the database: this part of the process took a few hours.

Our full corpus yielded 374,108 ngrams, with 123 characters; 125,384 ngrams occurred just once. This confirms what is already known about word hapax in documents: contrary to intuition, their proportion does not follow a decreasing curve, however large is the corpus. It holds here. Figure 2 shows the distribution of the ngrams by frequency order (first 100 frequencies), often said to follow Zipf law.

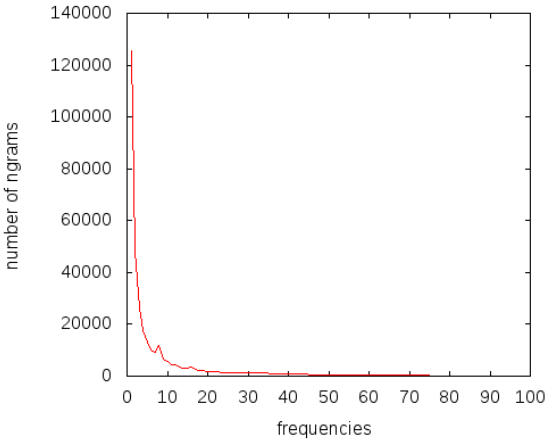

Figure 2: Ngram Distribution.

But the formula they follow is steeper than Zipf law, shown in figure 3.

The sparsity of the matrix depends on the threshold selected; the increase of the number of non-zero components follows astonishingly regular curves ; figure 4 shows the curve for our test corpus, figure 5 for the full one.

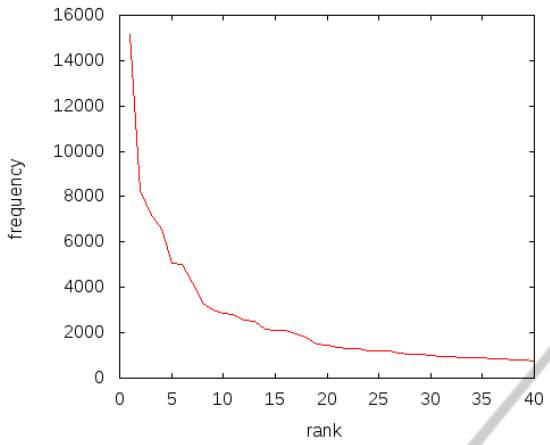

Figure 3: Zipf law on James Joice's Ulysses.

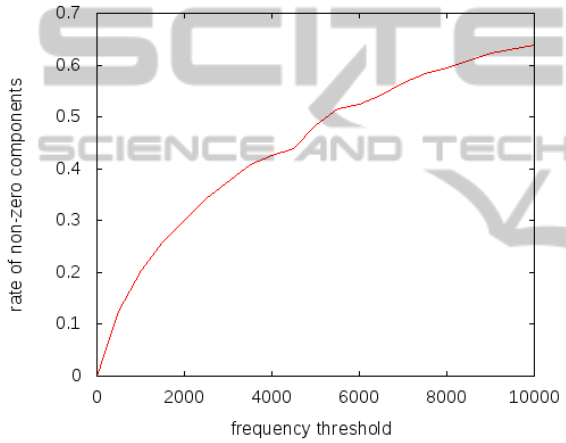

Figure 4: Matrix Sparsity – test corpus.

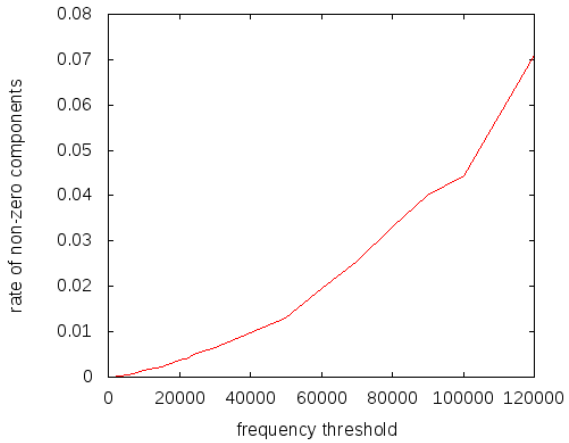

Figure 5: Matrix Sparsity – full corpus.

5.2 SOM Results

In our test corpus, a threshold of 10,000 yielded 73 candidates graphemes; only one 4-gram, ' de ', and six 3-grams ('de ' ' de' 'ait' 'es ' ' le' 'it '). In our full corpus, a threshold of 25,000 yielded 88 candidates, with the same 4-gram and ten 3-grams (adding 'le ' 'nt ' 'ent' ' qu'). These contains graphemes or grammatical markers (le, de). The exemples given below are taken from the full corpus with thresholds between 10,000 and 25,000.

Preliminary remarks: we tested the following SOM parameters: neighbourhood, topology, number of epochs and neighbourhood learning curve. The best results were with hexagonal topology (6 neighbours), a neighbourhood starting with 25% of the map and finishing with 0 (the winner itself), with a gaussian neighbourhood learning, and 30.000 epochs (if not we indicate it explicitly).

In nearly all results four classes of data emerge: (a) vowels (including vocalic graphemes); (b) consonants (including qu); (c) true syllables ("il ra la"); (d) false syllables: written syllables usually pronounced as consonants, mostly at the end of words ("re se te"). Then there are leading and trailing items (ex. ' a' is leading a, 'a,' or 'a ' are trailing a), mostly in categories (a, b, d).

Figures 6-8 show excerpts of a map 7*7, full corpus, 10.000 epochs, euclidian distance.

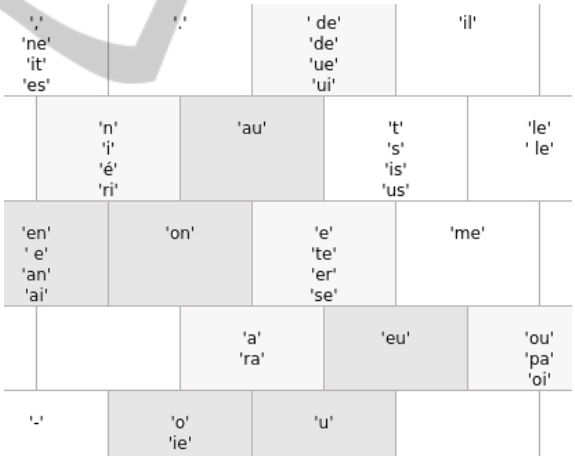

Figure 6: Vowels cluster.

Vowels, including vocalic graphemes, are located in one cluster (in grey neurons containing only vowels, light grey for vowels and others). Most nasal vocalic graphemes ("en an on in") are concentrated on three neurons ('in' is in the first neuron on the second line from the top, not visible here). Grapheme 'un' is far from the others (not in the excerpt), probably due to its being mostly used as an article, with spaces as context. One could argue that in those cases 'un' is not a phonetic grapheme, but one with pure morphological meaning, as 'à' or 'où' (which are not frequent enough to exceed the threshold). Leading 'e' is in one

of the central cluster, while ‘e’ (a mute vowel apart from initial position) is in a cluster containing “false syllables”.

Figure 7 has a very homogeneous consonant cluster (only ‘un’, apostrophe sign, and “hollow” ngram ‘e d’ are not consonants).

Figure 7: Consonants cluster.

Figure 8 shows the false syllable repartition in the same map. Leading ‘ le’ and ‘ de’ have been classified with normal ‘le’ and ‘de’ ; trailing ‘de ’ and isolated ‘ de ’ are classified together, farther away.

Figure 8: False syllables repartition.

True syllables do not form a cluster, they are dispersed in the whole map (as “ra pa ro il” here). But punctuation and spaces are grouped together in figure 9.

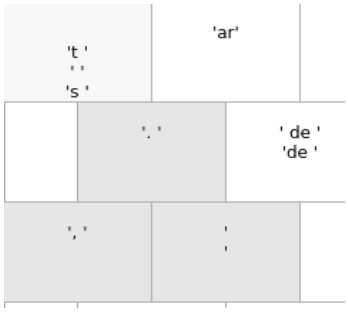

Figure 9: Punctuation cluster.

Vocalic and consonant clusters are to be found in most runs. False syllables can be encountered in more or less clear clusters; true syllables are rarely found in clusters. This holds even for very little maps, as the one presented in figure 10 (2*2, 30,000 epochs) on the left: neuron (0,0) false syllables, (0,1) trailing elements and punctuation, (1,0) consonants, (1,1) vowels (similar repartition on the right, with cosine similarity).

With a threshold of 10,000, yielding 255 candidates, the clearest category is punctuation, nearly always grouped in adjacent neurons without mixing. The consonant one is second to it, being in its own clusters. A bigger set of true syllables confuses a little the results for the other categories, and the nasal vocalic graphemes are not as clearly distributed as in the 22,000 threshold data.

The threshold of 19,000, selected to contain the grapheme “ch”, show very similar results to those already seen, with “ch” rightly incorporated in the consonant cluster. Not much difference either with the threshold of 25,000.

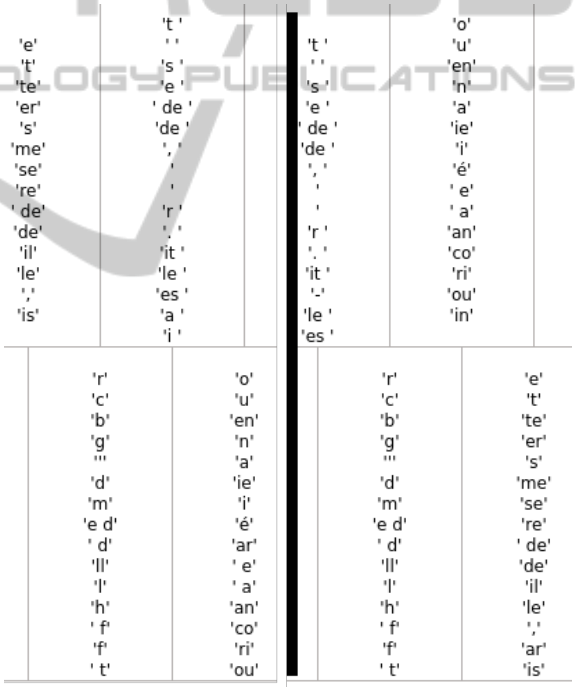

Figure 10: Complete maps, euclidian / cosine.

Cosine similarity works nearly as well as euclidian distance (see figure 10 for a comparison); as regards vocalic clusters, it performs slightly worse (separating clusters). Manhattan distance performs quite badly here, as it tends to group more data in less neurons and confuses the categories. So it seems SOM is immune to the curse of dimensionality.

6 CONCLUSIONS

The feasibility of a raw-unicode text-based approach for detecting at least phonemic properties of graphemes is illustrated by those results. Much still has to be done, as generalizing to other languages, especially poorly described ones. We focused here on the experimentation protocol, ensuring that no shortcut was taken which would violate our first constraint: do not put information into data that you want to get information from.

We began trials on the Gutenberg Project French utf-8 files (approx. 330 Mo), and are devising a strategy to be able to process it in reasonable time. As the results for our full corpus were much more to the point than the results for the test corpus, we hope that it will yield a highest rate of correct classification (getting rid of spurious data and of the influence of writer's style).

We have in immediate perspective an extension outside the Unicode Basic Multilingual Plane. But this will complexify the code, as low and high surrogates have each the size of a BMP Unicode character and the prefix trie must be modified to support that, which will modify its breadth (but not its depth).

An extension including Spectral Clustering (Ng et al., 2002) and Hidden Markov Model analyses is planned in a near future. Our aim is to be able to reproduce the experiments in (Goldsmith and Xanthos, 2009) and to permit the crossing of clustering with neural networks as proposed in (Bassani and Araujo, 2014).

REFERENCES

Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional spaces. In *8th International Conference on Database Theory - ICDT*.

Bassani, H. and Araujo, A. (2014). Dimension selective self-organizing maps with time-varying structure for subspace and projected clustering. In *IEEE Transactions on Neural Networks and Learning Systems*.

Bernard, G. (1997). Experiments on distributional categorization of lexical items with self organizing maps. In *WSOM'97 International Workshop on Self Organizing Maps. Espoo, Finland.*

Bernard, G. and Mariage, J.-J. (2005). Post-processing of grammatical patterns produced by self organizing maps. In *8th Conference on Pattern Recognition and Information Processing PRIP'05, Minsk, Belarus*.

Goldsmith, J. and Riggle, J. (2012). Information theoretic approaches to phonology: the case of finnish vowel harmony. *Natural Language & Linguistic Theory*, 30(3).

Goldsmith, J. and Xanthos, A. (2009). Learning phonological categories. *Language*, 85(1).

Guy, J. B. M. (1991). Vowel identification: An old (but good) algorithm. *CRYPTOLOGIA*, 15(3).

Harris, Z. (1968). *Mathematical Structures of Language*. Interscience Publishers New York, 1st edition.

Honkela, T., Lagus, K., and Kohonen, T. (1997). Websom: Self-organizing maps of document collections. *Neurocomputing*, 21.

Jones, M. and Mewhort, D. (2004). Case sensitive letters and bigram frequency counts from large-scale english corpora. *Behavior Research Method, Instruments & Computers*, 36(3).

Kohonen, T. (1995). *Self-Organizing Maps*. Springer, Berlin, 1st edition.

Mandelbrot, B. (1957). *Étude sur la loi d'Estoup et de Zipf : fréquence des mots dans le discours. Logique, langage et théorie de l'information*.

Mayer, R., Roigel, A., and Rauber, A. (2008). Map-based interfaces for information management in large text collection. *Journal of Digital Information Management*, 6(4).

Mayer, T. and Rohrdantz, C. (2013). Phonmatrix : Visualizing co-occurrence constraints of sounds.

Ng, A., Jordan, I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems 14*.

Petruszewycz, M. (1973). L'histoire de la loi estoup-zipf : documents. *Mathmatiques et Sciences Humaines*, 44.

Pham, M. and Lee, J.-L. (2015). Combining successor and predecessor frequencies to model truncation in brazilian portuguese.

Powers, D. (1997). Unsupervised learning of linguistic structure: an empirical evaluation. *International Journal of Corpus Linguistics*, 2.

Sukhotin, B. (1962). Experimental'noe vydelenie klassov bukv s pomoju evm. *Problemy strukturnoj lingvistiki*, 236.

Tsimboukakis, N. and Tambouratzis, G. (2007). Self-organizing word maps for context-based document classification. In *WSOM'07, International Workshop on Self-Organizing Maps*. Neuroinformatic Group Bielefeld, Germany.

Weingarten, R., Nottbusch, G., and Will, U. (2004). Morphemes, syllables and graphemes in written word production. *Multidisciplinary Approaches to Language Production*.