OPEN SOURCE ENGINEERING

openeering

A SCILAB PROFESSIONAL PARTNER

powered by ENGINSOFT

AN UNSUPERVISED TEXT CLASSIFICATION METHOD IMPLEMENTED IN SCILAB

Author: *Massimiliano Margonari*

Keywords. Scilab; Text classification; Self organizing maps.

Abstract: In this paper we present an unsupervised text classification method based on the use of a self organizing map (SOM). A corpus of roughly 200 plain text documents have been considered. Some Scilab scripts have been prepared to read and process these documents, train the neural network and graphically render the results.

Contacts m.margonari@openeering.com

1. Introduction

Text mining is a relatively new research field whose main concern is to develop effective procedures able to extract meaningful information - with respect to a given purpose - from a collection of text documents. There are many contexts where large amounts of documents have to be managed, browsed, explored, categorized and organized in such a way that information we are looking for can be accessed in a fast and reliable way. Let us simply think to the internet, which probably is the largest and the most used library we know today, to immediately understand why the interest around the text mining has increased so much during the last two decades.
A reliable document classification strategy can help in information retrieval, to improve the effectiveness of a search engine for example, but it can be also used to automatically understand if an e-mail message is spam or not.
Scientific literature proposes many different approaches to classify texts: it is sufficient to perform a web search to find out a large variety of papers, forums and sites discussing about this topic.
The subject is undoubtedly challenging for researchers who have to consider different and problematic aspects coming out when working with text documents and natural language. Usually texts are unstructured, they have different lengths and they are written in different languages. Different authors usually write on different topics, with their own styles, lexicons, vocabularies and jargons, just to highlight some issues. The same concept can be expressed in several different ways. As an extreme case, the same sentence can be graphically rendered in different ways:

You are welcome!

U @r3 w31c0m3!

This strategy can be used to cheat less sophisticated e-mail spam filters, which probably are not able to correctly categorize a received message and waste it. Some filters are based on simple algorithms which do not consider the real meaning of a message but they just look the single words inside, one at a time. The search of an exhaustive and exact solution to text mining problem is extremely difficult, or practically impossible.
Many mathematical frames have been developed for text classification: naïve Bayes classifiers, supervised and unsupervised neural networks, learning vector machines and clustering techniques are just a short - and certainly not complete - list of possible approaches which are commonly used in this field. They have both advantages and disadvantages. For example, some of them usually ensure a good performance but they have to be robustly trained in advance using predefined categories: other ones do not require a predefined list of categories, but they are less effective. For this reason the choice of a specific strategy is often tailored on the categorization problem that has to be solved.
In spite of their differences, all text categorization approaches have however a first common problem to solve: first text has to be processed to extract the main features contained inside. This operation erases the “superfluous” from documents, retrieving only the most relevant information: the categorization algorithm will therefore work only with a series of features characterizing the document. This operation has a fundamental role and it can be lead to unsatisfactory results if it has not been conducted in an appropriate way.
Another crucial aspect of data mining techniques is the postprocessing and summarization of results, which have to be read and interpreted by a user.
This means that the faster and the most effective data mining algorithm is useless if

improperly fed or if results cannot be represented and interpreted easily.
Our personal interest for these techniques was born some weeks ago when reading the last issue of the EnginSoft newsletter. In a typical newsletter issue there usually are many contributions of different kinds: there are papers presenting case studies coming from several industrial sectors, there are interviews, corporate and software news and much more. Sometimes there are also paper discussing “strange” topics, for the CAE community, as probably this one may seem to be.
A series of questions came out. Does the categorization used in the newsletter respect a real structure of documents or is it simply due to an editorial need? Can we imagine a new categorization based on other criteria? Can we discover categories without knowing them a-priori? Can we finally have a representation of this categorization? And finally, can we have a deeper insight into our community?
We decided to use the EnginSoft newsletters (see [3]) and extract all English articles, starting from the first issue up to the last one. In this way we built the “corpus”, as it is usually called by the text miners community, that is the set of text documents that have to be considered.
The first issues of the newsletter were quite completely written in Italian, but English contributions occupy the most of pages in the last years. This certainly reflects the international growth of EnginSoft. The corpus was finally composed by 248 plain text documents of variable lengths.
The second step we performed was to set up a simple text mining procedure to find out possible categorizations of corpus, taking into account two fundamental aspects. First we have to consider that we do not have any a-priori categorization. Second problem is that the corpus cannot be considered as “large” but, on the contrary, probably too poor to have clear and robust results.
We finally decided to use an unsupervised self organizing map (SOM) as a tool to discover possible clusters of documents. This technique has the valuable advantage to not require any predefined classification and certainly to allow a useful and easily readable representation of a complex dataset, through some two-dimensional plots.
The paper concludes with a funny application of a Markov chain, used in this case to automatically write apparently “meaningful” text.

Figure 1: This image has been generated starting from the text of the EnginSoft Flash of the Year 7 n°1 issue and the tool available in [3].

2. Preprocessing the corpus

It easy to understand that one of the difficulties that can arise when managing text, looking one-word-at-a-time and disregarding for simplicity all the aspects concerning lexicon, is that we could consider as "different" words which conceptually can have the same meaning. As an example, let us consider the following words which can appear in a text; they can be all summarized in a single word, such as "optimization":

optimization, optimizing, optimized, optimizes, optimization, optimality.

It is clear that a good preprocessing of a text document should recognize that different words can be grouped under a common root (also known as *stem*). This capability is usually obtained through a process referred to as *stemming* and it is considered fundamental to make text mining more robust. Let us imagine launching a web search engine with the keyword "optimizing". We probably would like that documents containing the words "optimization" or "optimized" are also considered when filling a results list. The question is that probably the real goal of our research is to find out all the documents where optimization issues are discussed.
The ability of associating a word to a root is certainly difficult to codify in a general manner. Also in this case there are many strategies available: we decided to use the *Porter stemming approach* (it is one of the most used stemming technique for processing English words: see paper in [5]) and apply it to all words composed by more than three letters.
If we preprocess the words listed above with the Porter stemming algorithm the result will be always the stem "optim". It clearly does not have any meaning (we cannot find "optim" in an English dictionary) but this does not represent an issue for us: we actually need "to name" in a unique way the groups of words that have the same meaning.
Another ability that a good preprocessing procedure should have is to remove the so-called *stop words,* that is, all words which are used to build a sentence in a correct way, according to the language rules, but they usually do not significantly contribute to determine the meaning of the sentence. Lists of English stop words are available on the web and they can be easily downloaded (see [2]): they contains words such as "and", "or", "for", "a", "an", "the", etc…
In our text preprocessor we decided to also insert a procedure that cuts out all the numbers, the dates and all the words made of two letters or less; this means that words such as "2010" or "21th" and "mm", "f", etc… are not considered. Also mathematical formulas and symbols are not taken into consideration.

3. Collect and manage information

The corpus has to be preprocessed to produce a sort dictionary, which collects all the stems used by the community; then, we should be able to find out all the most interesting information describing a document under exam to characterize it.
It's worth to mention that the dictionary resulting from the procedure described above using the EnginSoft newsletters is composed by around 7000 stems. Some of them are names, surnames and acronyms such as "CAE".
It immediately appears that a criterion to judge the importance of a stem in a document within a corpus. To this purpose we decided to adopt the so–called *tf-idf* coefficient, *term frequency – inverse document frequency*, which takes into account both the relative frequency of a stem in a document and the frequency of the stem within the corpus. It is defined as:

$$(tf - idf)_{w,d} = tf_{w,d} * idf_w$$

being

$$tf_{w,d} = \frac{n_{w,d}}{\sum_{k=1}^{N} n_{w,k}}$$

$$idf_w = ln \frac{N}{|1 + n_{w,C}|}$$

where the subscripts w and d stand for a given word and a given document respectively in the corpus C - done by N documents - while $n_{i,j}$ represents the number of times that the word i appears in the j-th document. This coefficient allows us to translate words into numbers.
In Figure 2 the corpus has been graphically represented, plotting the matrix containing the non-zero tf-idf coefficients computed for each stem, listed in columns, as they appear while processing the documents, listed in rows. The strange profile of the non-zero coefficients in the matrix is obviously due to this fact: it is interesting to see that the most used stems appear early on while processing documents, and that the rate of dictionary growth - that is the number of new stems that are added to the dictionary by new documents - tends to gradually decrease. This trend does not depend, on average, on the order used in document processing: the resulting matrix is always denser in the left part and sparser on the lower-right part. Obviously, the top-right corner is always void.
The matrix in Figure 2 represents a sort of database which can be used to accomplish a document search, according to a given criterion: for example, if we wanted to find out the most relevant documents with respect to the “optimization” topic, we should simply look for the documents corresponding to the highest tf-idf of the stem *optim*. The results of this search are collected in Table 1, where the first 5 documents are listed.

Document title	**Published in the Newsletter**	**tf-idf of stem “optim”**
The current status of research and applications in Multiobjective Optimization.	Year 6, issue 2	0.0082307
Multi-objective optimization for antenna design.	Year 5, issue 2	0.0052656
Third International Conference on Multidisciplinary Design Optimization and Applications.	Year 6, issue 3	0.0050507
modeFRONTIER at TUBITAK-SAGE in Turkey.	Year 5, issue 3	0.0044701
Optimal Solutions and EnginSoft announce Distribution Relationship for Sculptor Software in Europe.	Year 6, issue 3	0.0036246

Table 1: The results of the search for “optimization” in the corpus using the tf-idf coefficient.

In Table 2 we list the stem which register the highest and the lowest (non zero) tf-idf in the dictionary, together with the documents where they appear. More generally, it is interesting to see that high values of tf-idf are obtained by words that frequently appear in a short document, but that globally are not used at all (see the acronym “VPS”). On the contrary, low values of this coefficient are obtained by common words in the corpus (see “design”) that are infrequently used in long documents.
In Figure 3 the histogram of the tf-idf coefficient and the empirical cumulate density

function are plotted. It can be seen that the distribution is strongly left-skewed: this means that there are many stems that are largely used in the corpus, having in this way very low values of tf-idf. For this reason the logarithmic scale has been preferred to have a better representation of data.

Document title	Published in the Newsletter	Stem	tf-idf
VirtualPaintShop. Simulation of paint processes of car bodies.	Year 2, issue 4	VPS	**Max** 0.0671475
Combustion Noise Prediction in a Small Diesel Engine Finalized to the Optimization of the Fuel Injection Strategy	Year 7, issue 3	design	**Min (non-zero)** 0.0000261

Table 2: Stems with the maximum and the minimum (non zero) tf-idf respectively found in the corpus are reported in the table together with the document title where they appear.

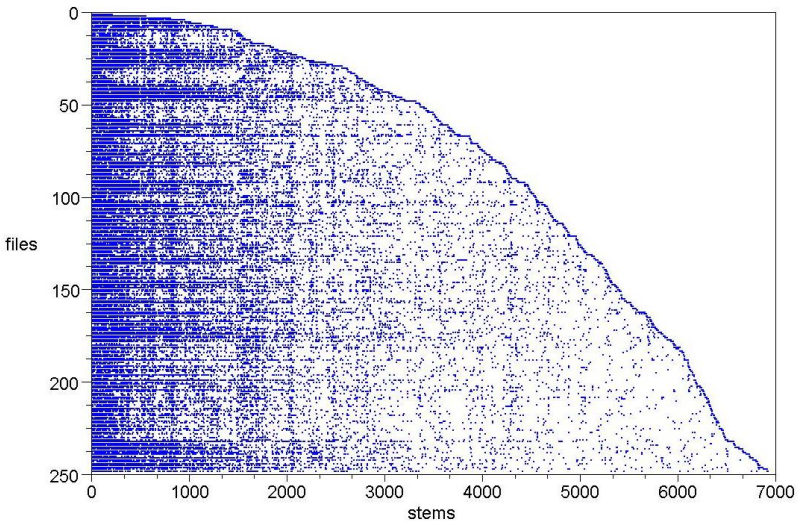

Figure 2: A matrix representation of the non-zeros tf-idf coefficients within the corpus. The matrix rows collect the text files sorted in the same order as they are processed, while the columns collect the stems added to the dictionary in the same order as they appear while processing the files.

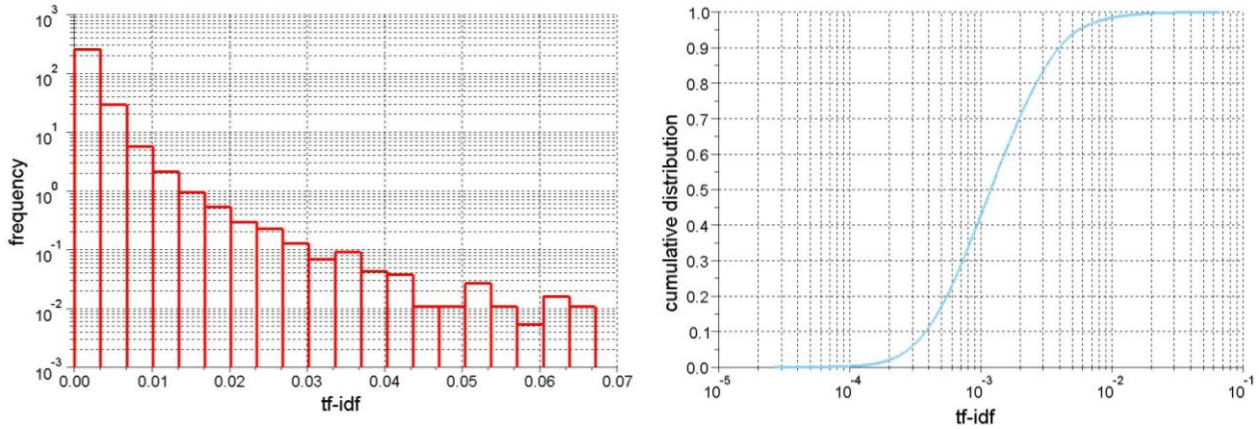

Figure 3: The histogram (left) and the empirical cumulative distribution (right) of the tf-idf. The distribution has clearly a high skewness: the large majority of stems has a low tf-idf. For this reason the logarithmic scale has been used in the graphs.

4. A text classification using Self Organizing Maps

The Self Organizing maps (SOMs) are neural networks which have been introduced by Teuvo Kohonen (see for example [6]). One of the most valuable characteristics of such maps is certainly the fact that they allow a two-dimensional representation of multivariate datasets, preserving the original topology; this simply means that the map does not alter the distances between records in the original space when projecting them in the two-dimensional domain. For this reason they can be used to navigate multidimensional datasets and to detect groups of records, if present. A second interesting characteristic of these maps is that they are based on an unsupervised learning: this is the reason why, sometimes, such maps are said to learn from the environment. They do not need any imposed categorization or classification of data to run, but they simply project the dataset "as it is". The mathematical algorithm behind this maps is not really difficult to understand and therefore to implement; however, the results have however to be graphically represented in such a way that they can be easily accessed by the user. This is probably the most difficult task when developing a SOM: fortunately Scilab has a large set of graphical functions which can be called to build complex outputs, such the one in Figure 6.
A common practice suggests using a honey-comb representation of the map, where each hexagon stands for a neuron: colors and symbols are used to draw a result (e.g. a dataset component or the number of records in a neuron).
The user has to set the dimensions of the map, choosing the number of neurons along the horizontal and the vertical directions (see Table 3, where the set up of our SOM is briefly reported) and the number of training cycles that have to be performed. Each neuron has a prototype vector (that is a vector with the same dimension of the designs in the dataset) which should be representative, once the net has been trained, of all the designs pertaining to that neuron. Certainly the easiest way to initialize the prototypes is to choose random values for all their components, as we did in our case.
The training consists of two phases: the first one is called "rough phase", the second one "fine tuning" and they usually have to be done with slightly different set-ups to obtain the best training, but operationally, they do not present any difference. During the training a design is submitted to the net and assigned to the neuron whose prototype vector is closest to the design itself; then, the prototypes of the neurons in the neighborhood are updated trough an equation which rules the strength of the changes according, for example, to the training iteration number and to the neuron distances.
During a training cycle all designs have to be passed to the net, always following a different order of submission to ensure a more robust training. There is a large variety of updating rules available in the literature which can be adopted according to the specific problem. We decided to use a Gaussian training function with a constant learning factor which is progressively damped with the iteration number. This leads to a net which progressively "freezes" to a stable configuration. This can be seen as the solution of a nonlinear projection problem of a multivariate dataset on a two dimensional space.
At the end of training phase, each design in the dataset has a reference neuron and each prototype vector should summarize at best the designs in their neuron. For this reason the prototype vectors can be thought as a "summary" of the original dataset and used to graphically render information through colored pictures.
One of the most frequent criticisms to SOMs that we hear within the engineering community is that these maps do not provide numbers but rather colored pictures that only "gurus" can interpret. We are pretty convinced that this is a wrong feeling; these maps, and consequently the colored pictures used to present results, are obtained with a precise algorithm such those used in other fields. As an example, let us remember that even

results coming from a finite element simulation of a physical phenomenon are usually presented through a plot (e.g.: stress, velocity or pressure fields in a domain) and that they can change as the model set up changes (e.g.: mesh, time integration step…) and that therefore they have to be always interpreted by a skilled engineer.
We submitted the dataset with the tf-idf coefficients and ran a SOM training with the setup summarized in Table 3. To avoid that stems with very high and too low values play a role in the SOM training, we decided to keep only those belonging to the interval $[0.0261 - 2.6484]\cdot 10^{-3}$: this interval has been chosen starting from the empirical cumulative distribution reported in Figure 3 and looking for the tf-idf corresponding to the 0.1 and the 0.8 probability respectively. In this way the extremes, which could be also due to typos, are cancelled out from the dataset, ensuring a more robust training. The dictionary decreases from 7000 to around 5000 stems, which are considered to be enough to describe exhaustively the corpus, keeping quite common words and preserving all the peculiarities of documents.
Once the SOM has been trained, we decided to use the “distance matrix” as the best tool to “browse” the results. The so-called D-matrix is a plot of the net where the color scale is used to represent the mean distance between the neurons’ prototype vector and their neighbors (red means “far”, blue means “close”). In this way one can understand how the dataset is distributed on the net, with just a glance, and also detect clusters of data, if any. This graphical tool can be also enriched with other additional information, plotted together with the color scale, giving the possibility to represent the dataset in a more useful way. An example of this enriched versions are given in Figures 5 and 6.

Grid		**Rough Phase**	**Fine Phase**
Number of horizontal neurons = 15	Training = sequential	nCycles = 50	nCycles = 10
Number of vertical neurons = 15	Sample order = random	iRadius = 4	iRadius = 1
Grid initialization = random	Learning factor = 0.5	fRadius = 1	fRadius = 1
Scaling of data = no	Training function = gaussian		

Table 3: The setup used for the SOM training phase.

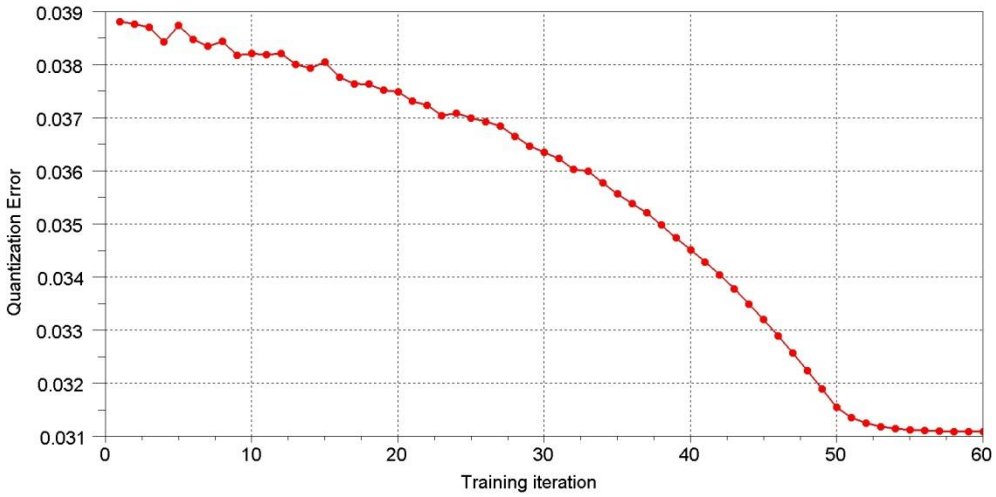

Figure 4: The quantization error plotted versus the number of the training iterations.

Looking at the plot of the D-matrix reported in Figure 5 one can conclude that there are mainly two large groups of papers (the two blue zones), which are not however sharply separated, and many outliers. It is not easy to identify in a unique way other clusters of

papers, being too high the distance between neurons' prototypes outside the blue zones.
The dimension of the white diamonds superimposed to the neurons is proportional to the number of documents which pertains to the neuron. It is clear that there are many files that fall in one of these two groups.
Looking to the map drawn in Figure 6, we can try to understand what is the main subject discussed by papers in these group. We actually decided to report the stems which gain the highest tf-idf in the prototype vectors, providing in this way two "keywords" that identify papers falling in the neurons. In the first group, positioned on the left-upper part of the map, certainly there are documents discussing about EnginSoft and the international conference. Documents discussing about optimization and computational fluid dynamics belong to the second group, positioned on the central-lower part of the net, actually stems such as "optim" and "cfd" often gain the highest tf-idf.
It is interesting to see that some important relations and links appear in the net. For example, the lower-right corner is occupied by documents mainly discussing about laminates and composite materials. Going up in the net, following the right border, we meet papers on casting and alloys. On the top corner, contributions by our Turkish partner, Figes, have found place. Moving to the left we meet stems such as "technet", "allianc" and "ozen" that remember us the great importance that EnginSoft gives to the international relationships and to the "net". We can also find several times "tcn", "cours" and "train", which can be certainly due to the training activities held and sponsored by EnginSoft in the newsletter. In the upper left corner the stem "race" can be found: the competition corner - we could say - because contributions coming from the world of races (by Aprilia, Volvo and others) fall here.
Figure 6 certainly gives us a funny but valuable view on our community.
Another interesting output which can be plotted is certainly the position that documents written by an author assume in the net. This could be useful to detect common interests between people of a numerous community. This kind of output is summarized in Figure 7, where, starting from left to right, the position of documents by Stefano Odorizzi, by Akiko Kondoh and by Silvia Poles are reported. It can be seen that our CEO contributions, the "EnginSoft Flash" at the beginning of all the issues, fall in the first group of documents, where EnginSoft and its activities are the focus.
Akiko's contributions are much more spread on the net: some of them fall in the left-lower portion that could be viewed as the Japanese corner, some other between the two main groups. Finally, we could conclude that Silvia's contributions mainly focus on PIDO and multi-objective optimization topics.
In Figure 8 shows the prototype vector of a neuron in the first group of documents. On the right side of the same picture, the first 10 stems which register the highest values of tf-idf are reported. These stems could be read as keywords that concisely define documents falling in the neuron.

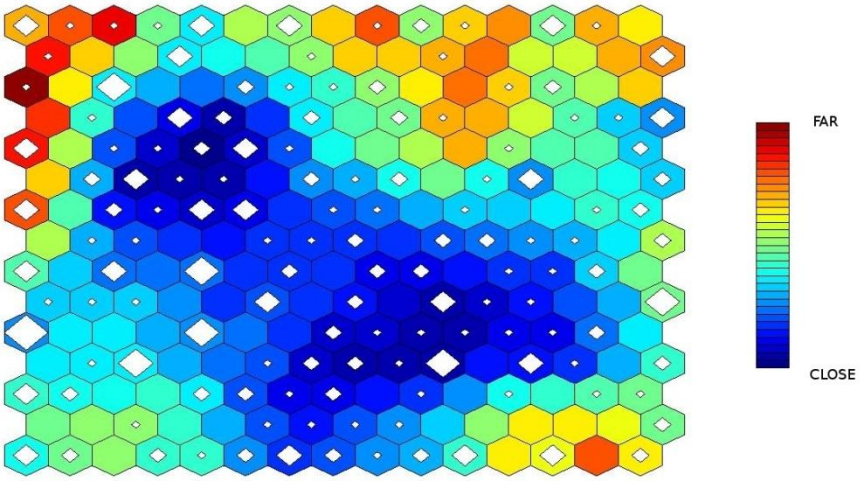

Figure 5: The D-matrix. The white diamonds give evidence of the number of files pertaining to the neuron. Two groups of documents (blue portions) can be easily detected.

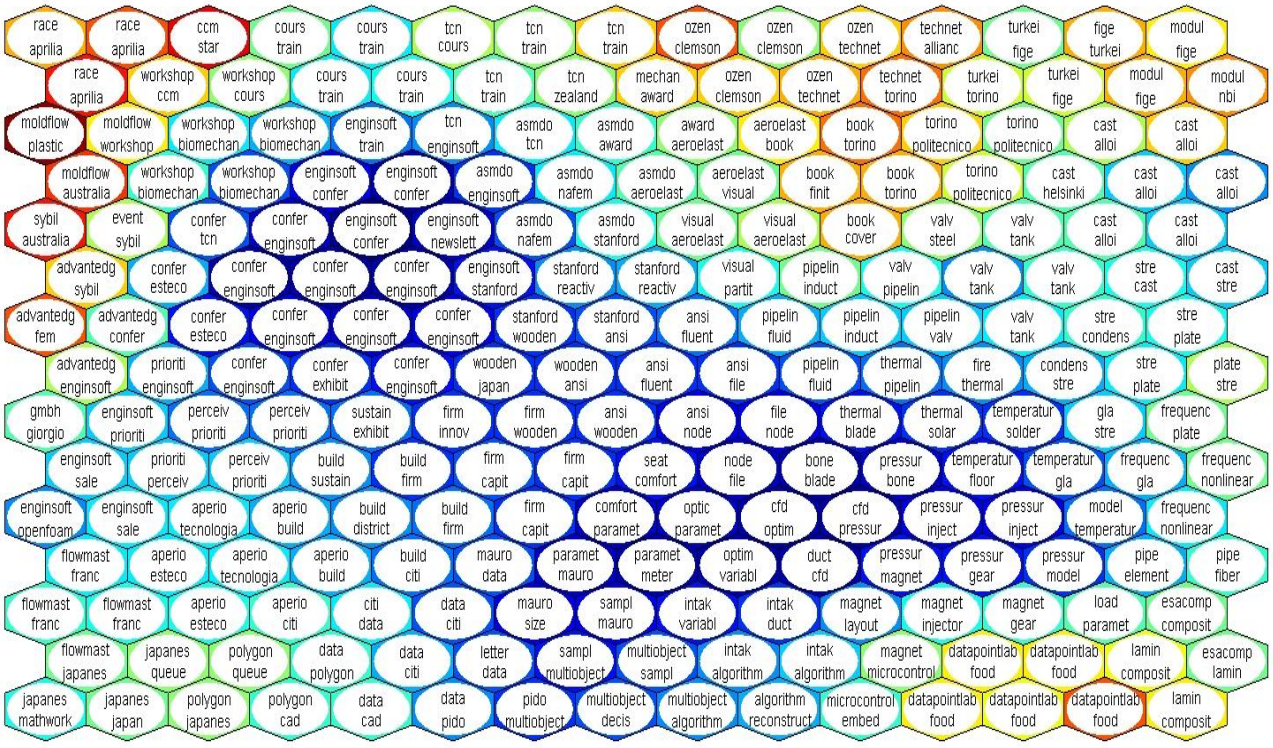

Figure 6: The D-matrix. For each neuron the first two stems with highest tf-idf as given by the prototype vectors are reported, in the attempt to highlight the main subject discussed by articles falling in the neurons.

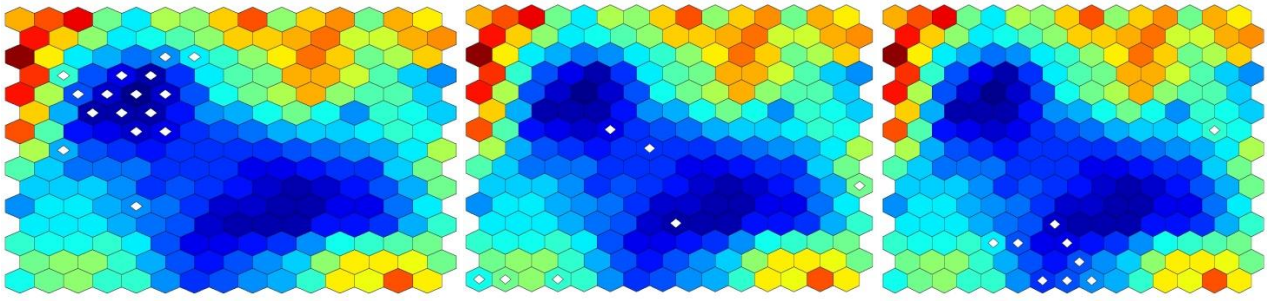

Figure 7: The contributions by Stefano Odorizzi (left), by Akiko Kondoh (middle) and by Silvia Poles (right) as they fall in the SOM (see white diamonds).

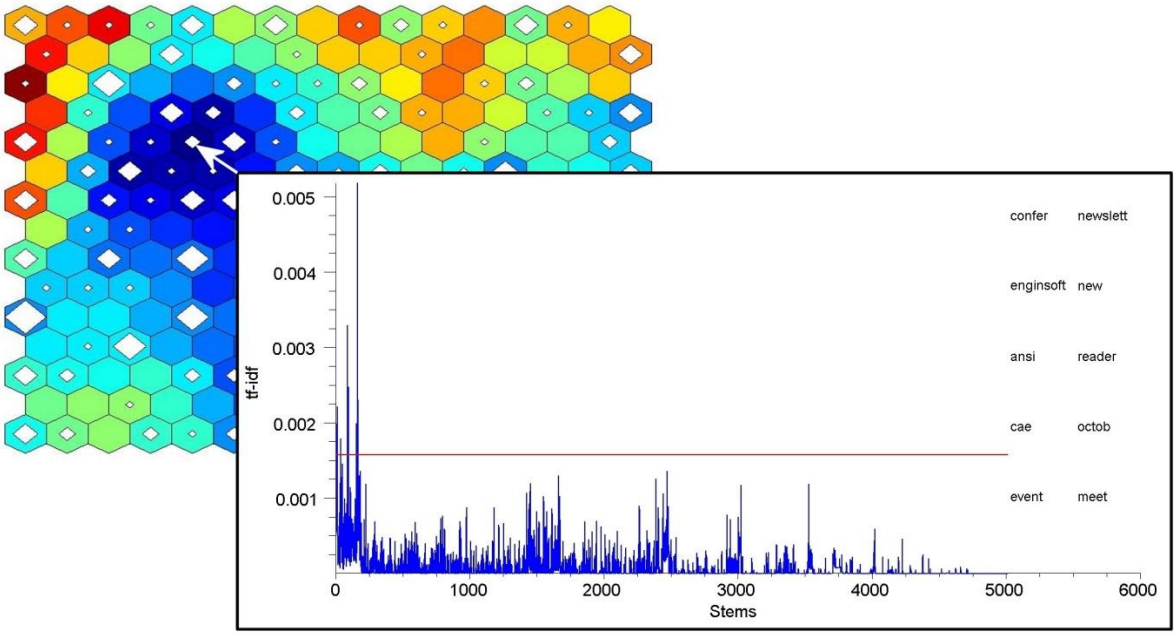

Figure 8: The prototype vector of the pointed neuron in the net: the tf-idf is plotted versus the stems in the dictionary. On the right the first 10 highest tf-idf stems are displayed. The horizontal red line gives the lowest tf-idf registered by the 10th stem.

5. An ergodic Markov chain writes *meaningful* text

To conclude this work we decided to play with words. We implemented a simple Markov chain to randomly generate text which should have, at a first glance, some meaning. This is another technique which could be used to cheat the spam filters which automatically tries to understand meaning of messages. Actually, the generated text sounds, in some portions at least, as a natural language text: an accurate reading done by a human being however can unmask the trick easily.
Markov chains are known to be an extremely powerful and versatile mathematical theory used in many fields and due to Andrey Markov, a Russian mathematician. In our case, the basic idea is to train a net whose nodes are words, coming out from a reference text. The "weight" of a connection is related to the probability that the node (word) is followed by the connected node (word). These weights are computed during the training of the chain and they reflect *in some sense* the writing style.
Once the training has been completed the chain can be run automatically, just choosing a starting node; it is simply necessary to randomly choose the next node (word) according to the probability associated to the node connections. In this way, it is possible to generate always different texts which can seem to have a certain meaning. Obviously, the result strongly depends on the training text, which should be long enough and rich of different words.
We developed a Scilab procedure that uses a Markov chain to this aim. We collected all the "EnginSoft Flash" by our CEO in a single text file to produce the training set. Finally, we started the chain to randomly generate a text; a result can be read as follows:

The international conference and peace, standard and developments and products including car running on the finite element methods. Besides his 20 years, and science, the world congress, it is the ansys italian conference and our knowledge at berkeley, at the simulation, and projects and technology providers in april we encourage our call for our

knowledge in its realisation.
Engineers enterprises and system modelling study of the expected spin-off enginsoft france showcased its development prospects for computational technologies and related readings. Moreover, it has identified a busy time include an event for example, manufacturing, for a whole team of the congress will publish research labs. Taking request for enginsoft, modefrontier users' meeting to welcome training on-line courses on the world's leading cae/cad model deformation and the positive conclusion of biomechanics, their views on cae and that enginsoft and coordinator.

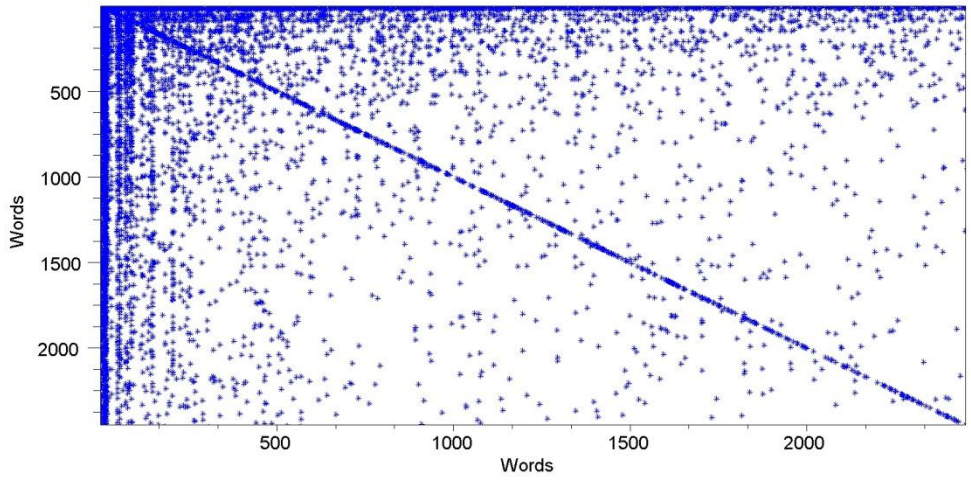

Figure 9: The Markov chain can be thought as a sparse matrix where rows and columns are the words in the dictionary and the stored values are the weight of the links.

6. Conclusions

In this work we have presented a text mining approach mainly based on the use of a self organizing map. We have considered the English articles published on old issues of the EnginSoft newsletters and preprocessed them adopting some well-known methodologies in the field of text mining.
The dataset has been used to train a self organizing map: results have been graphically presented and some consideration on the documents set have been proposed. The work ends with a funny implementation of a Markov chain which automatically and randomly generates text, which, at a first glance, could appear as meaningful.
All this work has been performed using Scilab scripts, expressly written for this aim.

7. References

[1] *http://www.scilab.org/* to have more information on Scilab.
[2] *http://www.ranks.nl/resources/stopwords.html* to have an exhaustive list of the English stop words.
[3] *http://newsletter.enginsoft.it/* to download the pdf version of the EnginSoft newsletters.
[4] *http://www.wordle.net/* to generate funny images starting from text.
[5] *http://tartarus.org/~martin/PorterStemmer/def.txt*
[6] *http://www.cis.hut.fi/teuvo/*