Using a co-similarity approach on a large scale text categorization task

Clément Grimal — Gilles Bisson

Université Joseph Fourier / Grenoble 1 / CNRS
Laboratoire LIG
Bat. CE4 – Allée de la Palestine
38610 GIERES
{Clement.Grimal, Gilles.Bisson}@imag.fr

ABSTRACT. *This paper presents a framework we developed for the second Large Scale Hierarchical Text Categorization challenge LSHTC2. The main idea is to propose a method allowing to deal with the terms variability among the categories in order to be able to find similarities between collections of documents belonging to the same category but having few common terms. Thus, we used a co-similarity based approach, named χ-Sim, that we introduced in previous work. Nevertheless, as this co-similarity methods are not highly scalable, we need to implement a "divide and conquer" approach to split the categories into a set of clusters containing semantically related documents. This lead to a two-stage strategy for the document categorization: first, we decide in which cluster the test document belongs, and then inside the elected cluster, we perform the final categorization that is based on our co-similarity approach.*

RÉSUMÉ. *Ce papier présente une architecture développée pour participer au second défi LSHTC de classification de textes à grande échelle. L'idée est de proposer une méthode permettant de traiter la variabilité terminologique des documents afin de trouver des similarités entre des collections appartenant à la même catégorie sémantique, mais n'ayant que peu de termes en communs. Nous avons utilisé une approche basée sur la co-similarité, nommé χ-Sim, présentée dans de précédents travaux. Néanmoins, cette méthode de calcul de co-similarité passant difficilement à l'échelle, nous avons défini une approche de type « diviser pour régner » pour découper les catégories en groupes (clusters) contenant des documents sémantiquement proches. Ceci nous conduit à une stratégie à deux étapes pour la tâche de classification des documents : premièrement, nous affectons chaque document à un cluster, puis à l'intérieur de celui-ci, nous réalisons la classification finale basée sur notre approche de calcul de co-similarité.*

KEYWORDS: *Categorization, text mining, large-scale database*

MOTS-CLÉS : *Classification, fouille de texte, masse de données*

1. Introduction

With the development of the web, and the high availability of the storage spaces, more and more documents become accessible. The downside is that one can feel overwhelmed by the great amount of information available, and being able to navigate within such network of documents is a great challenge. One classical way to organize such information is to associate one or several category to a set of related documents, these categories being organized into a hierarchy or more generally a directed graph. In many popular databases such as DMOZ or Wikipedia, the choice of the category labels is done by the users. Not only this task is very time consuming, but also it leads inevitably to major variability and many inconsistencies in the labelling. Here, the main problem to tackle is to deal with the huge number of categories in the databases: for instance, the International Patent Classification (IPC) contains about 70,000 categories and in Wikipedia there exist more than 20,000 categories linked together by different kinds of relations. Thus, category labelling of large databases is a great challenge for the machine learning and for the information retrieval communities, and we need to develop new algorithms able to scale up well for large Category Systems.

The use of pre-existing taxonomies of categories (related to documents) is a very natural way to bring *a priori* information to help the categorization of new documents, but this information is not always used. Indeed, one can split the main categorization methods into three categories:

– *Big bang approaches* (or knowledge poor) try to solve directly the problem by using a single classifier working on all categories at the same time. However, tests reported in [MAD 07] show that in this paradigm the processing time, both in learning and in test, is huge. That said, the learning method recently developed in [BOT 08] allows to improve this kind of approach.

– *Top-down approaches* (or knowledge intensive), divide the initial problem into simpler problems by using the hierarchy of categories. Although more efficient than big-bang methods from a computational point of view, the top-down approach faces a major problem which is the propagation of the labelling errors from the most general clusters (top of the hierarchy) to the most specific ones (final categories). One can note that it is also possible to automatically infer relations between the categories, by a clustering method for instance, in order to be able to divide the problem afterwards.

– *Hybrid approaches* [XUE 08] try to combine the strengths of the two previous strategies: first, the hierarchy of categories is used to divide the database into collection of homogeneous sub-problems then, a different classifier is learned on each of these sub-problems.

The Pascal Challenge on Large Scale Hierarchical Text Classification (second edition, the first one was in 2010) offers several categorization tasks, involving different documents and category hierarchies. We must notice that in such large collections of text documents, it is very likely to observe a huge vocabulary variability as 1) numerous authors participated to the writing of the documents, and 2) across long periods of time. Therefore, without terminological knowledge, in many cases, it can be difficult to find a link between a test document and a training document, even if both deal with

the same topic and belong to the same categories, if their authors have chosen different vocabulary. Facing this problem, it seems necessary to set up a method allowing to determine if two words are similar (maybe synonyms) or not. Unfortunately, in this challenge, all terms have been replaced by unique numerical identifiers, thus linguistic approaches are not helpful.

In previous work [HUS 10b], we developed in the frame of the *co-clustering*, a statistical algorithm allowing to compute a co-similarity, named χ-Sim, between documents and terms of a corpus. We make use of the duality between words and documents (each one can be seen as a descriptor for the other), as well as their respective higher-order co-occurrences. For example, let us consider a training corpus belonging to the category *oceanography* in which a subset of documents contains a significant number of co-occurrences between the words *sea* and *waves* and another subset in which the words *ocean* and *waves* co-occur. So, we can infer that the worlds *ocean* and *sea* are conceptually related even if they do not directly co-occur in any document. Such a relationship between *waves* and *ocean* (or *sea* and *waves*) is termed as a first-order co-occurrence and the conceptual association between sea and ocean is called a second-order relationship. This concept can be generalized to higher-order (3rd, 4th, 5th, etc) co-occurrences. By using this approach, we will be able to find some similarities between two documents of the same category even if they are using a different terminology. Hence, we (hope to) solve the vocabulary variability problem.

However, the time and space complexities of our co-similarity approach are respectively in $\mathcal{O}(n^3)$ or $\mathcal{O}(n^2)$, preventing any direct use of our method on a huge collection of documents. For instance, for the DMOZ dataset it is impossible to store a terms similarity matrix for the complete vocabulary (almost 600K) since this kind of matrices are not sparse. Consequently, to learn the classifier, we use an Hybrid approach as described previously. In a first step, using the information provided by the categories hierarchy, we group together the documents that are semantically related, into a smaller collection of so called *clusters*. Thus, in a second step, we use our co-similarity approach for every cluster of documents, as their vocabulary becomes tractable. Moreover, we add two word selection stages in order to only work with the terms that offer the best discrimination among categories. Finally, the categorization process takes place in two steps: first, assigning the test document to one (or more) cluster of training documents, then find the most similar documents with the help of the local term similarity matrix computed during the training process. Then, every of these training documents votes for its own categories.

The rest of this paper is organized as follows. In Section 2, the word selection steps, along with the gathering of the documents into clusters, are described. Section 3 presents the co-similarity approach we used in order to learn the local term similarity matrices. In Section 4, we describe the categorization process. Finally, we discuss the (very preliminary) experiments in Section 5.

2. The pre-processing phase

Here we describe the overall architecture (Figure 1) of the preprocessing step needed to create the learning sets that will be used by χ-Sim. First, after having selected a subset of the most discriminant words for each category L_k, we split the initial set of documents into a collection of clusters $\{C_1, ..., C_p\}$ containing semantically related documents; this stage is describe in section 2.1. Secondly, for each cluster C_i, we build two vectors of relevant words: the inter-cluster words W_{C_i} and the intra-cluster W_{S_i}. The vector W_{S_i} contains a selection of the terms of C_i that are discriminant (for a given criteria) with respect to the other clusters $C_{j\neq i}$, and the vector W_{S_i} contains a selection of terms of C_i allowing to discriminate between the categories occurring in C_i. This two stage are described in sections 2.2 and 2.3.

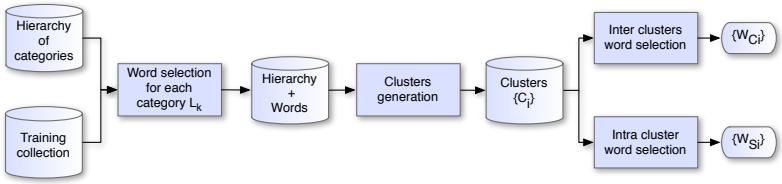

Figure 1. *The pre-processing phase.*

2.1. *Creation of the clusters*

In order to evaluate the relevant co-similarities between words of a given collection of documents, we need to ensure that these documents contain a large subset of semantically related words; in other terms, these documents must concern similar topics. The simplest way to achieve that is to trust the hierarchy of categories of each database, and to consider that documents belonging to the same subtree are semantically related. Of course, the closer we are from a bottom category (i.e. a leaf), the higher similarity the documents will (probably) have.

However, a question arise: what is the size of the subtree we have to consider to create the clusters? To use χ-Sim, the number of documents in each cluster must be large enough (about 100 documents) to be able to capture some interesting statistical similarities between the words, and at the same time it must be small enough to reduce the time and space complexity of the calculus. The best ratio is about one thousand documents per clusters. Unfortunately, the distribution of the documents within the categories is highly uneven in all the databases of the challenge. For instance, in DMOZ (edition 2011 of the challenge) the biggest category contains more than 5K documents and almost 20% of the categories just contain one or two documents. So, we need to define a strategy to deal with these two extreme situations:

– When a category is too large we used a sampling approach to reduce the number of documents to 800. Currently, this selection is done randomly, but we could use

a better method: for instance, to run the *Partitioning Around Medoids* algorithm and keep the medoids as cluster representatives.

– When a category is too small, we merge the documents of this category with those of the closest neighbours. Here, the fusion process is guided by the hierarchy of categories and, at each step, we only merge the sibling categories. Finally, each cluster is characterized by the highest common parent of the categories it contains.

More generally, if we don't have a hierarchy of categories, the merging step can be done by using any clustering algorithm. Besides, building automatically such hierarchy can be also helpful if one decides to use a multi-level categorization process as we do, but with more than two levels.

However, in all cases, we need a criteria to decide when a set of sibling categories must be clustered together. The idea is to use an iterative, bottom-up, approach in which we merge the most similar categories first. However, in order to avoid computing a similarity matrix between all the sibling documents of each category, we built up an intermediate representation of the data, the idea being:

– For each inner category, to build a vector of the words offering the best discrimination rate with respect to its sibling categories.

– To base the evaluation of the sibling similarity on the similarities between the documents and their direct parent category (one if the hierarchy of categories is strict, several if we have a DAG structure).

2.1.1. *Characterization of the categories*

Thus, in order to efficiently compare two nodes of the categories hierarchy, we had to build a vector of the representative words for every node L_k, only selecting the terms that offer the best discrimination among all the nodes. Numerous work [YAN 97, FOR 08] compare different ways to compute a score between a feature t and a category L_k. Here are three classical approaches among many others:

– the Mutual Information: $\mathrm{I}(t, L_k) = \log \frac{P(t \wedge L_k)}{P(t)P(L_k)}$

– the Bi-Normal Separation: $\mathrm{BNS}(t, L_k) = \left| F^{-1}\left(\frac{P(t \wedge L_k)}{P(L_k)}\right) - F^{-1}\left(\frac{P(t \wedge \neg L_k)}{P(\neg L_k)}\right)\right|$ where F^{-1} is the inverse Normal cumulative distribution function, as commonly available from statistical tables.

– the double conditional probabilities: $\mathrm{CP}(t, L_k) = P(t|L_k)P(L_k|t)$

In the experiments we conducted, we elected the double conditional probabilities as for the DMOZ dataset of the challenge of 2010, it provided the best prediction results. For a given category, after having computed the score for every single word, there is two main different approach [NOV 04]:

– selecting the *best individual features*, i.e. considering that the features are independent the one from the others;

– or using a *sequential selection approach* that take note of the dependencies between words, e.g. we start by selecting the best individual feature, and then we update the scores of the others, in order to avoid taking two highly correlated features.

However, the computation required to perform such a sequential selection is prohibitive with a large vocabulary and we had to use a best individual features method, however the second approach is clearly more accurate.

To decide the number of words to keep to characterize a category, we used a threshold based on the ratio of the sum of the score of the already selected words, over the sum of the scores of all the words. For instance, if the criteria is the Mutual Information, this corresponds to selecting enough words to have 80% of the total Mutual Information. Using this simple idea, we allow the selection algorithm to keep very few words, as soon as they offer a high discrimination ratio. Thus we reduce both space and time complexity for the following steps of our process.

After having selected the "best" words for all the leaves categories, we pass them on to their parents, and perform the same step on these nodes, doing so until every node of the hierarchy has been processed, in order to be able to compare any pair of nodes of the hierarchy.

2.1.2. *Merging of the categories*

Now, having built the representative words for every category (corresponding to the node of the hierarchy), we can run the clusters creation algorithm described by the pseudo-code algorithm 1.

Algorithm 1 Clusters creation

Require: MAXC, categories hierarchy
Ensure: clusters
  clusters ← [leaves categories]
  candidates ← [parents of categories]
  while |clusters|>MAXC **do**
    winner ← $\underset{c \in \text{candidates}}{\text{argmax}} \frac{\sum_{c' \in \text{children}(c)} \text{size}(c') \times \text{Cosine}(c, c')}{\sum_{c' \in \text{children}(c)} \text{size}(c')}$
    append winner to cluster and remove it from candidates
    append parents of winner to candidates and remove them from clusters
  end while

We begin the algorithm by initializing the set of clusters C_i by creating one cluster for each leaf category of the hierarchy. We start with about 28,000 clusters for the DMOZ dataset. We also initialize the set of candidate categories, which are all the direct parent nodes of the leaves categories. As we want to build clusters of documents that are as semantically related as possible, at each step of the algorithm, we compute a score for all the categories that are candidate to become a new cluster (see Fig. 2).

For a given candidate node, this score is the weighted mean of the Cosine similarity between itself and its children nodes. The weights are the size of the children nodes and the Cosine is computed by using the vectors of representative words built in the previous step. Finally, when the number of clusters is smaller than the MAXC parameter (user-defined), the aggregation process is over and the last step consists in sampling the documents of the clusters that contains too many documents (the threshold for sampling is also a user-defined parameter). Indeed, as we said earlier, to use

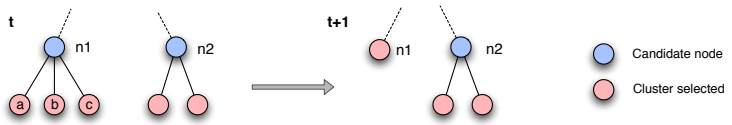

Figure 2. *The main step of the clusters creation algorithm. Here the node n_1 is more similar to its children than node n_2, so it becomes a new cluster and the previous ones* a, b, c *are removed.*

χ-Sim, the number of documents must be small enough to reduce the time and space complexity of the computation.

For the two datasets extracted from Wikipedia, the hierarchies are no longer simple tree structures, since the categories can have more than one parent. This can be problematic when a node n with more than one parent must be remove from the list of clusters. We made the choice to remove n from the cluster list only if all its parents are already contained in clusters. In a nutshell, the parent categories have the "shared custody" of their children.

2.2. *Finding the inter-clusters words*

During the final categorization process, in order to predict to which cluster belongs a test document, we need to compute the similarity between this test document and all the clusters. Again, we will use a cosine measure between the vector of terms of the document and a vector of terms characterizing each cluster. Thus, we need to perform another word selection step on all the words occurring in a cluster. More precisely, for every cluster C_i, we perform this word selection step to obtain the set of words W_{C_i} that will be used to represent it. It is worth noticing that *we cannot re-use* the previously selected terms as described in Sec. 2.1. Indeed, in the previous step the goal was to find words that offer the best discrimination among all the different categories, whereas now, we want to base the similarity measure on the terms providing the best discrimination rate among the different selected clusters.

As described in Sec. 2.1 (Characterization of the categories), again we use a threshold for dynamically selecting the number of words. Figure 3 plots, for the DMOZ dataset (2010 edition), the average number of words selected and the variation of Accuracy (The ratio of the number of documents assigned to the cluster they effectively belong to, over the total number of training documents) against the threshold. By varying this threshold from 10% to 100%, we observe that the gain in precision is not very huge when the threshold is larger than 50%. Hence, using this threshold can further reduce the number of terms to keep, allowing to decrease the complexity of the similarity calculus.

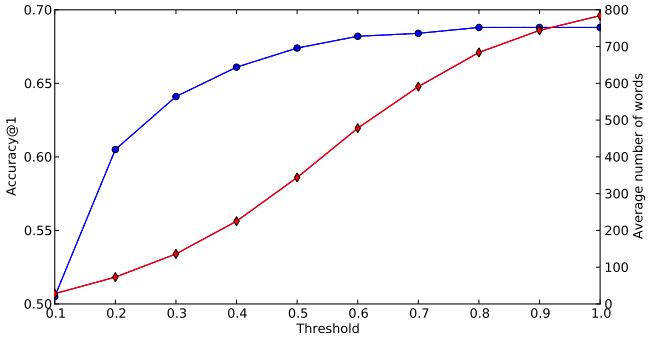

Figure 3. *For the DMOZ dataset (2010 edition), average number of words selected by cluster (diamond) and Accuracy@1 (circle) against the threshold for the inter-clusters words selection step.*

2.3. *Finding the intra-clusters words*

Finally, in order to be able to efficiently perform our learning process based on the χ-Sim co-similarity measure, at the intra-cluster level, we first need to perform a very last word selection step. Here, we want to select the terms offering the best discrimination rate among the different categories within a cluster, with respect to the double conditional probabilities criteria. This selection is performed by using the same ideas as described in Sec. 2.1.

Again, we cannot re-use the terms previously selected, as for now, the goal is not to find the words that offers the best discrimination among all the other categories (see Sec. 2.1), neither among the other clusters (see Sec. 2.2), but among the other categories contained within the cluster. Thus, for a given cluster C_i, we select the "best" words for every category it contains, and then build the union of all these terms to obtain W_{S_i}, the local vocabulary of the cluster that will be used both in the learning phase and in the categorization phase.

3. The learning phase

The goal of our learning phase is to compute, for every cluster C_i, the similarity between all the pairs of the intra-cluster words W_{S_i}, that have been selected inside a cluster as described in Sec. 2.3. Learning such similarities will enable us to find links between documents written by different authors that use different vocabularies. This section presents the χ-Sim algorithm, that was originally developed for text clustering, and that has been successfully extended for text categorization [HUS 10a].

We will use the following classical notations: matrices (in capital letters) and vectors (in small letters) are in bold and all variables are in italic:

– *Data matrix:* let $\mathbf{M}$ be the matrix representing a cluster having r documents (rows) and c words (columns); m_{ij} corresponds to the 'intensity' of the link between

the i^{th} document and the j^{th} word; $\mathbf{m}_{i:} = [m_{i1} \cdots m_{ic}]$ is the row vector representing the document i and $\mathbf{m}_{:j} = [m_{1j} \cdots m_{rj}]$ is the column vector of the word j.

– *Similarity matrices:* $\mathbf{SR}$ and $\mathbf{SC}$ represent the square and symmetrical row similarity and column similarity matrices of size $r \times r$ and $c \times c$ respectively, with $\forall i, j = 1..r, sr_{ij} \in [0, 1]$ and $\forall i, j = 1..c, sc_{ij} \in [0, 1]$.

– *Similarity function:* function $\mathrm{F_s}(\cdot, \cdot)$ is a generic function that takes two elements m_{il} and m_{jn} of $\mathbf{M}$ and returns a measure of the similarity $\mathrm{F_s}(m_{il}, m_{jn})$ between them.

3.1. *The χ-Sim measure*

Usually, the similarity (or distance) measure between two documents $\mathbf{m}_{i:}$ and $\mathbf{m}_{j:}$ is defined as a function – denoted here as $\mathrm{Sim}(\mathbf{m}_{i:}, \mathbf{m}_{j:})$ – that is more or less the sum of the similarities between words occurring in both $\mathbf{m}_{i:}$ and $\mathbf{m}_{j:}$:

$$\mathrm{Sim}(\mathbf{m}_{i:}, \mathbf{m}_{j:}) = \mathrm{F_s}(m_{i1}, m_{j1}) + \cdots + \mathrm{F_s}(m_{ic}, m_{jc}) \tag{1}$$

Now let's suppose we already know a matrix $\mathbf{SC}$ whose entries provide a measure of similarity between the words of the corpus. In parallel, let's introduce, by analogy to the norm L_k (Minkowski distance), the notion of a *pseudo-norm* k. The main idea is to generalize (1) in order to take into account all the possible pairs of words occurring in documents $\mathbf{m}_{i:}$ and $\mathbf{m}_{j:}$. In this way, we "capture" not only the similarity between their common words but also the similarity coming from words *that are not directly shared* by the two documents. Of course, for such terms, we weight their contribution to the document similarity sr_{ij} by their own similarity sc_{ln}. Thus, the overall similarity between documents $\mathbf{m}_{i:}$ and $\mathbf{m}_{j:}$ is defined in (2):

$$\mathrm{Sim}^k(\mathbf{m}_{i:}, \mathbf{m}_{j:}) = \sqrt[k]{\sum_{l=1}^{c} \sum_{n=1}^{c} \left(\mathrm{F_s}\left(m_{il}, m_{jn}\right)\right)^k \times sc_{ln}} \tag{2}$$

Assuming that $\mathrm{F_s}(m_{il}, m_{jn}) = m_{il} \times m_{jn}$, as for the Cosine similarity, we can rewrite (2) as:

$$\mathrm{Sim}^k(\mathbf{m}_{i:}, \mathbf{m}_{j:}) = \sqrt[k]{\left(\mathbf{m}_{i:}\right)^k \times \mathbf{SC} \times \left(\mathbf{m}_{j:}^{\mathrm{T}}\right)^k}$$

where $\left(\mathbf{m}_{i:}\right)^k = \left[\left(m_{ij}\right)^k \cdots \left(m_{ic}\right)^k\right]$ and $\mathbf{m}_{j:}^{\mathrm{T}}$ the transpose of $\mathbf{m}_{j:}$.

Finally, we want to map the similarity measure to $[0, 1]$, so we need to introduce a normalization factor to do so. Investigating extensions of the Generalized Cosine measure, we decided to use the following normalization:

$$sr_{ij} = \frac{\mathrm{Sim}^k(\mathbf{m}_{i:}, \mathbf{m}_{j:})}{\sqrt{\mathrm{Sim}^k(\mathbf{m}_{i:}, \mathbf{m}_{i:})} \times \sqrt{\mathrm{Sim}^k(\mathbf{m}_{j:}, \mathbf{m}_{j:})}} \tag{3}$$

However, this normalization is what we will call a *pseudo-normalization* since it guaranties that $sr_{ii} = 1$, but it does not satisfy that $\forall i, j \in 1..r, sr_{ij} \in [0, 1]$. A counter example is given in [HUS 10b], but it is nevertheless interesting to investigate the results one can obtain from varying k, including values lower than 1, as suggested by [AGG 01] for the norm L_k, to deal with high dimensional spaces.

3.2. *A Generic χ-Sim Algorithm*

Equation (3) allows us to compute the similarity between two documents. The formula to compute the similarity between two words is similar. The extension over all pair of documents and all pairs of words can be generalized under the form of a simple matrix multiplication. We need to introduce a new notation here, $\mathbf{M}^{\circ_k} = \left((m_{ij})^k\right)_{i,j}$ which is the element-wise exponentiation of $\mathbf{M}$ to the power of k. The algorithm follows:

1) We initialize the similarity matrices $\mathbf{SR}$ (documents) and $\mathbf{SC}$ (words) with the identity matrix $\mathbf{I}$, since, at the first iteration, only the similarity between a document (resp. a word) and itself equals 1 and zero for all other documents (resp. words). We denote these matrices as $\mathbf{SR}^{(0)}$ and $\mathbf{SC}^{(0)}$.

2) At iteration t, we calculate the new similarity matrix between documents $\mathbf{SR}^{(t)}$ by using the similarity matrix between words $\mathbf{SC}^{(t-1)}$:

$$\mathbf{SR}^{(t)} = \mathbf{M}^{\circ_k} \times \mathbf{SC}^{(t-1)} \times \left(\mathbf{M}^{\circ_k}\right)^{\mathrm{T}} \text{ and } sr_{ij}^{(t)} \leftarrow \frac{\sqrt[k]{sr_{ij}^{(t)}}}{\sqrt[2k]{sr_{ii}^{(t)} \times sr_{jj}^{(t)}}}$$

We do the same thing for the words similarity matrix $\mathbf{SC}^{(t)}$:

$$\mathbf{SC}^{(t)} = \left(\mathbf{M}^{\circ_k}\right)^{\mathrm{T}} \times \mathbf{SR}^{(t-1)} \times \mathbf{M}^{\circ_k} \text{ and } sc_{ij}^{(t)} \leftarrow \frac{\sqrt[k]{sc_{ij}^{(t)}}}{\sqrt[2k]{sc_{ii}^{(t)} \times sc_{jj}^{(t)}}}$$

3) We set to 0 the $p\,\%$ of *the lowest similarity values* in the similarity matrices $\mathbf{SR}$ and $\mathbf{SC}$. This step is used to efficiently reduce the impact of the noise in the similarities and is described in [HUS 10b].

4) Steps 2 and 3 are repeated t times (typically 4 iterations are enough) to iteratively update $\mathbf{SR}^{(t)}$ and $\mathbf{SC}^{(t)}$.

It is worth noting here that even though χ-Sim computes the similarity between each pair of documents using all pairs of words, the overall complexity of the algorithm remains comparable to classical similarity measures like Cosine. Given that – for a generalized matrix of size n by n – the complexity of matrix multiplication is in $\mathcal{O}(n^3)$ and the complexity to compute $\mathbf{M}^{\circ_n}$ is in $\mathcal{O}(n^2)$, the overall complexity of χ-Sim is given by $\mathcal{O}(tn^3)$.

4. The categorization phase

The categorization is based on a two-level strategy. For a given test document, we first assign it to the k most similar clusters, then for each cluster, to the k' most similar documents. At the clusters level, we chose to compute the similarities between the test document and the clusters C_i with a classical Cosine measure. this measure is compute between the word vector of the document and the vector W_{C_i} selected during the pre-processing phase. Then, we decide to keep the k most similar clusters for the second level of the categorization process. Then, at the intra-clusters level, for each of the selected k clusters, we evaluate the similarities between the test document and all the documents of the cluster using the local similarity matrix $\mathbf{SC}$ computed during

the learning phase, using (3). In each of these k clusters, we keep the k' most similar documents, so we have selected a total of $k \times k'$ documents to decide the categories (a document can belong to several categories) of the test document (Fig. 4). This is achieved through a classical vote procedure: each of these documents votes for its own categories and we categorize the test document with the winning categories. The number of categories to keep is equal to the median number of categories occurring in the $k \times k'$ documents.

Figure 4. *The two-stage prediction process.*

5. Experiments

The framework has been implemented in Python, and only tested on the dry-run DMOZ dataset of the 2010 edition of the challenge. This hierarchy of categories for this dataset contains 2387 nodes, of which 1139 where leaves, i.e. labels for the documents. The training set contains 4463 documents, whereas the valid set and the test set respectively contain 1859 and 1857 documents. The organizers of the challenge used various evaluation measures but we compare our results with the ones published on the website of the challenge (`http://lshtc.iit.demokritos.gr/node/23`) using only the accuracy, which simply is the percentage of correctly labelled documents.

Table 1. *Results on the dry-run DMOZ dataset of the 2010 edition of the challenge. For our method, we give the number of clusters obtained, the accuracy obtained on the first step of our classification process on the training set, the global accuracy for the training and the validation sets. Final column is the accuracy on the test set of the best published method and of ours.*

	#clusters	Acc@1	Train Acc.	Valid Acc.	Test Acc.
Best published					46.8%
Our method	91	66.5%	66.3%	28.3%	28.3%

The training time for our algorithm was 10 minutes and the test time was 3 minutes, on an Intel Xeon @ 2.66GHz. Having a closer look at the 66.3% accuracy on the training set, it appears that most of our errors are due to an error at the clusters prediction level, as 66.5% of the training are affected to the correct cluster. Thus, this is an ongoing work and we need to refine the criteria used during the agglomerative approach to improve the results.

6. Conclusion and future works

We develop a generic framework for large-scale categorization using a co-similarity approach. Our approach can make use of a hierarchy of the categories if available,

in order to divide the set of training documents in consistent clusters as proposed in [XUE 08]. Thus, the main contribution of this paper is the integration of a co-similarity approach at the intra-clusters level that can be very useful to link semantically related documents written by many authors using different vocabulary, which is often the case in such large collections of text documents.

In the future, we will run tests on the different tasks of the challenge. We would like to test other words selection methods as it is required at three different levels of our process, being able to select better words could significantly increase the precision of our categorization system. Besides, so far, we only used our co-similarity approach at the intra-cluster level, but we plan to use it at the inter-cluster level as well, by computing a words similarity matrix for the inter-cluster terms, thus improving the performance of the first stage of our categorization phase. It would be also interesting to generalize our two-level strategy to a multi-level strategy.

Acknowledgements

This work is partially supported by the French ANR project FRAGRANCES under grant 2008-CORD 00801. Many thanks to the PASCAL team for setting up the LSHTC challenge.

7. References

[AGG 01] AGGARWAL C. C., HINNEBURG A., KEIM D. A., "On the Surprising Behavior of Distance Metrics in High Dimensional Space", *Lecture Notes in Computer Science*, Springer, 2001, p. 420–434.

[BOT 08] BOTTOU L., BOUSQUET O., "Learning using large datasets", *Mining Massive DataSets for security*, , 2008, Citeseer.

[FOR 08] FORMAN G., "BNS feature scaling: an improved representation over tf-idf for svm text classification", *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, New York, NY, USA, 2008, ACM, p. 263–270.

[HUS 10a] HUSSAIN S. F., BISSON G., "A supervised Approach to Text Categorization using Higher Order Co-Occurrences", *Society for Industrial and Applied Mathematics International Conference on Data Mining (SDM 2010)*, Columbus, Ohio, April 29-May 1 2010.

[HUS 10b] HUSSAIN S. F., GRIMAL C., BISSON G., "An Improved Co-Similarity Measure for Document Clustering", *ICMLA*, 2010.

[MAD 07] MADANI O., GREINER W., KEMPE D., SALAVATIPOUR M., "Recall systems: Efficient learning and use of category indices", *Proc. of AISTATS*, Citeseer, 2007.

[NOV 04] NOVOVIČOVÁ J., MALÍK A., PUDIL P., "Feature selection using improved mutual information for text classification", *Structural, Syntactic, and Statistical Pattern Recognition*, , 2004, p. 1010–1017, Springer.

[XUE 08] XUE G., XING D., YANG Q., YU Y., "Deep classification in large-scale text hierarchies", *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, 2008, p. 619–626.

[YAN 97] YANG Y., PEDERSEN J. O., "A Comparative Study on Feature Selection in Text Categorization", *ICML*, 1997, p. 412-420.