Computers in Human Behavior 30 (2014) 689–697

Contents lists available at ScienceDirect

Computers in Human Behavior

journal homepage: www.elsevier.com/locate/comphumbeh

Text classification using a few labeled examples

Francesco Colace [a,*,1], Massimo De Santo [a,1], Luca Greco [a,1], Paolo Napoletano [b,1]

[a] DIEM - Department of Information Engineering, Electrical Engineering and Applied Mathematics University of Salerno, 84084 Fisciano, Italy
[b] Department of Informatics, Systems and Communication University of Milan, Bicocca, 20126 Milan, Italy

ARTICLE INFO

Article history:
Available online 23 August 2013

Keywords:
Text mining
Text classification
Term extraction
Probabilistic topic
Model
Data mining

ABSTRACT

Supervised text classifiers need to learn from many labeled examples to achieve a high accuracy. However, in a real context, sufficient labeled examples are not always available because human labeling is enormously time-consuming. For this reason, there has been recent interest in methods that are capable of obtaining a high accuracy when the size of the training set is small.

In this paper we introduce a new single label text classification method that performs better than baseline methods when the number of labeled examples is small. Differently from most of the existing methods that usually make use of a vector of features composed of weighted words, the proposed approach uses a structured vector of features, composed of weighted pairs of words.

The proposed vector of features is automatically learned, given a set of documents, using a global method for term extraction based on the Latent Dirichlet Allocation implemented as the Probabilistic Topic Model. Experiments performed using a small percentage of the original training set (about 1%) confirmed our theories.

1. Introduction

With the proliferation of blogs, social networks and e-commerce sites, there is a great interest in supervised and semi-supervised text classification methods to reveal user sentiments and opinions (Magdalini Eirinaki & Pisal, 2012; Palus, Bródka, & Kazienko, 2011), to discover and classify the health service information obtained from the digital health ecosystems (Hai Dong & Hussain, 2011; Karavasilis, Zafiropoulos, & Vrana, 2010) and to classify web resources for improving the quality of web searches (Rahat Iqbal, 2012; Adam Grzywaczewski, 2012; Liu, 2006; Colace, Santo, Greco, & Napoletano, 2013).

The problem of supervised *text classification* has been extensively discussed in literature and metrics and measures of performance confirm that all the existing techniques achieve a high accuracy when trained on large datasets (Christopher, SchŸtze, & Manning, 2008; Sebastiani, 2002; Lewis, Yang, Rose, & Li, 2004).

However, most times, a supervised classifier is unfeasible in a real context, because a large labeled training set is not always available. It has been demonstrated that human employes about 90 min to label 100 documents. This makes the labeling task, for large datasets, practically unfeasible (McCallum, Nigam, Rennie, & Seymore, 1999; Ko & Seo, 2009).

Furthermore, the accuracy of classifiers, learned from a reduced training set (for instance made of hundreds instead of thousand of labeled examples), is quite low, around 30% (Ko & Seo, 2009). The low accuracy depends on the fact that most of the existing methods usually use a vector of features composed of weighted words that are obtained through the "bag of words" assumption (Christopher et al., 2008). Due to the inherent ambiguity of language (polysemy etc.), vectors of weighted words are insufficiently discriminative, especially when the classifier learns common patterns from a few labeled examples made of numerous features (Clarizia, Colace, De Santo, Greco, & Napoletano, 2011; Napoletano, Colace, De Santo, & Greco, 2012).

In this paper we demonstrate that a more complex vector of features, based on weighted pairs of words, is capable of overcoming the limitations of simple structures when the number of labeled samples is small. Specifically, we propose a linear single label supervised classifier that is capable, based on a *vector of features* composed of weighted pairs of words, of achieving a better performance, in terms of accuracy, than existing methods when the size of the training set is about 1% of the original and composed of only positive examples. The proposed vector of features is automatically extracted from a set of documents $\mathcal{D}$ using a global method for term extraction based on the Latent Dirichlet Allocation (Blei, Ng, & Jordan, 2003) implemented as the Probabilistic Topic Model (Griffiths, Steyvers, & Tenenbaum, 2007).

To confirm the discriminative property of the proposed features, we have evaluated the performance through a comparison with different methods using vectors of weighted words. The

* Corresponding author.
E-mail addresses: fcolace@unisa.it (F. Colace), desanto@unisa.it (M. De Santo), lgreco@unisa.it (L. Greco), paolo.napoletano@disco.unimib.it (P. Napoletano).
[1] The authors contributed equally to this work.

0747-5632/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.chb.2013.07.043

results, obtained on the top 10 classes of the ModApte split from the Reuters-21578 dataset, show that our method, independently of the topic, is capable of achieving a better performance.

2. Background and related works

2.1. Background

Following the definition introduced in Sebastiani (2002), a supervised *Text Classifier* may be formalized as the task of approximating the unknown target function $\Phi : \mathcal{D} \times \mathcal{C} \rightarrow \{T, F\}$ (namely the expert) by means of a function $\hat{\Phi} : \mathcal{D} \times \mathcal{C} \rightarrow \{T, F\}$ called the *classifier*, where $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_{|\mathcal{C}|}\}$ is a predefined set of *categories* and $\mathcal{D}$ is a set of *documents*. If $\Phi(\mathbf{d}_j, \mathbf{c}_i) = T$, then $\mathbf{d}_j$ is called a positive example (or a member) of $\mathbf{c}_i$, while if $\Phi(\mathbf{d}_j, \mathbf{c}_i) = F$ it is called a negative example of $\mathbf{c}_i$.

The categories are just symbolic labels: no additional knowledge (of a procedural or declarative nature) of their meaning is usually available, and it is often the case that no metadata (such as e.g. publication date, document type, publication source) is available either. In this paper we consider the classification task accomplished only on the basis of the knowledge extracted from the documents themselves (namely *endogenous knowledge*).

In practice, we consider a corpus $\Omega = \{\mathbf{d}_1, \ldots, \mathbf{d}_{|\Omega|}\} \subset \mathcal{D}$ of documents pre-classified under $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_{|\mathcal{C}|}\}$. The values of the total function Φ are known for every pair $(\mathbf{d}_j, \mathbf{c}_i) \in \Omega \times \mathcal{C}$.

We consider the initial corpus to be split into two sets, not necessarily of equal size:

1. the *training* set: $\Omega_r = \{\mathbf{d}_1, \ldots, \mathbf{d}_{|\Omega_r|}\}$. The classifier Φ for the categories is inductively built by observing the characteristics of these documents;
2. the *test* set: $\Omega_e = \{\mathbf{d}_{|\Omega_r|+1}, \ldots, \mathbf{d}_{|\Omega|}\}$, used for testing the effectiveness of the classifiers.

Here we consider the case of *single-label* classification, also called *binary*, in which, given a category $\mathbf{c}_i$, each $\mathbf{d}_j \in \mathcal{D}$ must be assigned either to $\mathbf{c}_i$ or to its complement $\bar{\mathbf{c}}_i$. In fact, it has been demonstrated that, through transformation methods, it is always possible to transform the multi-label classification problem either into one or more single-label classification or regression problems (Sebastiani, 2002; Tsoumakas & Katakis, 2007).

It means that we consider the classification under $\mathcal{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_{|\mathcal{C}|}\}$ as consisting of $|\mathcal{C}|$ independent problems of classifying the documents in $\mathcal{D}$ under a given category $\mathbf{c}_i$, and so we have $\hat{\phi}_i$, for $i = 1, \ldots, |\mathcal{C}|$, classifiers. As a consequence, the whole problem in this case is to approximate the set of function $\Phi = \{\phi_1, \ldots, \phi_{|\mathcal{C}|}\}$ with the set of $|\mathcal{C}|$ classifiers $\hat{\Phi} = \{\hat{\phi}_1, \ldots, \hat{\phi}_{|\mathcal{C}|}\}$.

2.2. Related works

A broad survey of a wide variety of text classification methods may be found in Sebastiani (2002), Yang and Liu (1999),ggarwal and Zhai (2012) and several of the discussed techniques have also been implemented and are publicly available through multiple toolkits such as the BoW toolkit (McCallum, 1996), Mallot (McCallum, 2002), WEKA,[2] and LingPipe.[3] As deeply discussed in these survey papers cited above, several key methods of text classification exist. In the following we enumerate some of them.

The *decision trees* method are essentially base on a hierarchical decomposition of the (training) data space, in which a predicate or a condition on the attribute value is used in order to divide the data space hierarchically (Quinlan, 1986). The hierarchical division of the data space is designed in order to create class partitions which are more skewed in terms of their class distribution. For a given text instance, the algorithm determines the partition that it is most likely to belong to, and use it for the purposes of classification.

Differently, *pattern (rule)-based* classifiers determine the word patterns which are most likely to be related to the different classes. The classifier constructs a set of rules, in which the lefthand side corresponds to a word pattern, and the right-hand side corresponds to a class label. These rules are used for the purposes of classification (Liu, Hsu, Ma, & Chen, 1999).

Support Vector Machines classiers attempt to partition the data space with the use of linear or non-linear delineations between the different classes (Cortes & Vapnik, 1995). The key in such classifiers is to determine the optimal boundaries between the different classes and use them for the purposes of classification.

Neural Network classifiers are used in a wide variety of domains for the purposes of classification (Bishop, 1995). This classifiers are related to SVM classifiers, because they both are in the category of discriminative classifiers, which are in contrast with the generative classifiers (Ng & Jordan, 2002). Moreover, simple neural networks are a form of linear classifiers, since the function computed by a set of neurons is essentially linear.

Bayesian classifiers (also called generative classifiers) attempt to build a probabilistic classifier based on modeling the underlying word features in different classes. The idea is then to classify text based on the posterior probability of the documents belonging to the different classes on the basis of the word presence in the documents (McCallum & Nigam, 1998).

Moreover, almost all classifiers can be adapted to the case of text data. Some of the other classifiers include *nearest proximity-based* classifiers (for instance neighbor classifiers), and *genetic algorithm-based* classifiers (Salton & McGill, 1983).

3. Document representation and dimension reduction

Texts cannot be directly interpreted by a classifier, therefore an indexing procedure that maps a text $\mathbf{d}_j$ into a compact representation of its content must be uniformly applied to the training and test documents. For the sake of simplicity we consider the case of the training set but the procedure described here is repeated in the case of test set.

Each document can be represented, following the *Vector Space Model* (Christopher et al., 2008), as a vector of term *weights*

$$\mathbf{d}_j = \{w_{1j}, \ldots, w_{|\mathcal{T}|j}\},$$

where $\mathcal{T}$ is the set of *terms* (also called *features*) that occur at least once in at least one document of Ω_r, and $0 \leqslant w_{nj} \leqslant 1$ represents how much term t_n contributes to a semantics of document $\mathbf{d}_j$.

If we choose to identify terms with words, we have the *bags of words* assumption, that is $t_n = v_n$, where v_n is one of the words of a vocabulary. The *bags of words* assumption claims that each w_{nj} indicates the presence (or absence) of a word, so that the information on the position of that word within the document is completely lost (Christopher et al., 2008).

To determine the weight w_{nj} of term t_n in a document $\mathbf{d}_j$, the standard tf–idf (*term frequency–inverse document frequency*) function can be used (Salton & McGill, 1983), defined as:

$$\text{tf}-\text{idf}(t_n, \mathbf{d}_j) = N(t_n, \mathbf{d}_j) \cdot \log \frac{|\Omega_r|}{N_{\Omega_r(t_n)}} \tag{1}$$

where $N(t_n, \mathbf{d}_j)$ denotes the number of times t_n occurs in $\mathbf{d}_j$, and $N_{\Omega_r(t_n)}$ denotes the document frequency of term t_n, i.e. the number of documents in Ω_r in which t_n occurs.

[2] http://www.cs.waikato.ac.nz/ml/weka/.
[3] http://alias-i.com/lingpipe/.

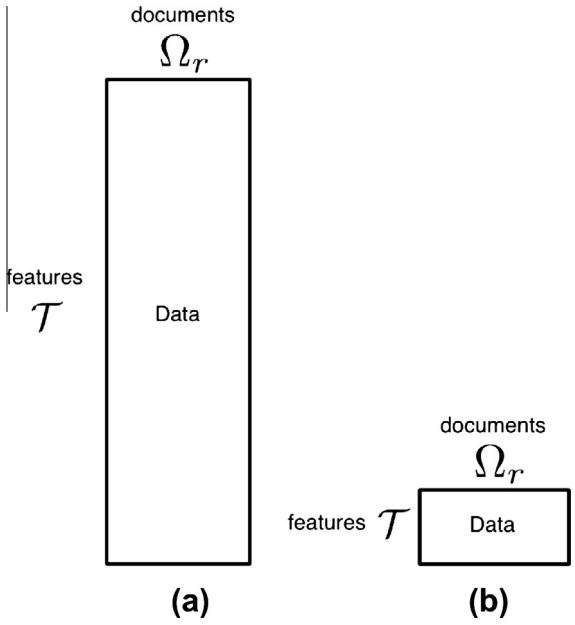

Fig. 1. Features-documents matrix. (a).In this case the number of features is much higher than the number of examples ($|\mathcal{T}| \gg |\Omega_r|$). (b).In this case $|\mathcal{T}| \ll |\Omega_r|$.

In order for the weights to fall in the [0,1] interval and for the documents to be represented by vectors of equal length, the weights resulting from tf–idf are usually normalized by cosine normalization, given by:

$$w_{nj} = \frac{\text{tf}-\text{idf}(t_n, \mathbf{d}_j)}{\sqrt{\sum_{n=1}^{|\mathcal{T}|}(\text{tf}-\text{idf}(t_n, \mathbf{d}_j))^2}} \quad (2)$$

In this paper, before indexing, we have performed the removal of function words (i.e. topic-neutral words such as articles, prepositions, and conjunctions) and we have performed the stemming procedure[4] (i.e. grouping words that share the same morphological root).

Once the indexing procedure has been performed, we have a matrix $|\mathcal{T}| \times |\Omega_r|$ of real values instead of the training set Ω_r, see Fig. 1(a). The same procedure is applied to the test set Ω_e.

3.1. Dimension reduction

Usually, machine learning algorithms are susceptible to the problem named the *curse of dimensionality*, which refers to the degradation in the performance of a given learning algorithm as the number of features increases. The increasing of the number of dimensions of the feature space causes the increasing of data sparsity so causing the lacking of statistic significance of data (Blum & Langley, 1997).

From a statistical point of view, in the case of supervised learning, it is desirable that the number of labeled examples in the training set should significantly exceed the number of features used to describe the dataset itself. In this way, even cases of sparsity can be overcame.

In the case of text documents the number of features is usually high, and it should be higher than the number of documents. In Fig. 1(a) we show the case of a training set composed of 100 documents and about 20,000 features obtained following the data preparation procedure explained in the previous paragraph. As you can see, $|\mathcal{T}| \gg |\Omega_r|$ while it is desirable to have the opposite condition, that is $|\mathcal{T}| \ll |\Omega_r|$, as represented in Fig. 1(b).

To deal with these issues, dimension *reduction techniques* are applied as a data pre-processing step or as part of the data analysis to simplify the whole data set (*global* methods) or each document (*local* methods) of the data set. As a result we can identify a suitable low-dimensional representation for the original high-dimensional data set, see Fig. 1(b).

In literature, we distinguish between methods that *select* a subset of the existing features or that *transform* them into a new reduced set of features. Both classes of methods can rely on a supervised or unsupervised learning procedure (Blum & Langley, 1997; Sebastiani, 2002; Christopher et al., 2008; Fodor, 2002; Berkhin, 2006; Noam & Naftali, 2001):

1. *Feature selection*: $\mathcal{T}_s$ is a subset of $\mathcal{T}$. Examples of this are methods that consider the selection of only the terms that occur in the highest number of documents, or the selection of terms depending on the observation of information-theoretic functions, among which we find the *DIA association factor*, *chi-square*, *NGL coefficient*, *information gain*, *mutual information*, *odds ratio*, *relevancy score*, *GSS coefficient* and others.
2. *Feature transformation*: the terms in $\mathcal{T}_p$ are not of the same type as the terms in $\mathcal{T}$ (e.g. if the terms in $\mathcal{T}$ are words, the terms in $\mathcal{T}_p$ may not be words at all), but are obtained by combinations or transformations of the original ones. Examples of this are methods that consider generating, from the original, a set of "synthetic" terms that maximize effectiveness based on *term clustering*, *latent semantic analysis*, *latent dirichlet allocation*, *principal component analysis* and others. After a transformation we could need to reduce the number of the new features through a selection method thus obtaining a new set $\mathcal{T}_{sp}$ that is a subset of $\mathcal{T}_p$.

In this paper we have used a *global* method for *feature extraction* that considers pairs of words instead of single words as basic features, and so we have obtained a new space $\mathcal{T}_p$ of features. The dimensionality of such a new space is very high: $\propto |\mathcal{T}|^2$. For this reason we need to reduce the transformed space in order to obtain a new space $\mathcal{T}_{sp}$ such that $|\mathcal{T}_{sp}| \ll |\mathcal{T}_p|$.

The method used to select the most representative pairs of words is based on the *Latent Dirichlet Allocation* (Blei et al., 2003) implemented as the *Probabilistic Topic Model* (Griffiths et al., 2007) and this is the core of the proposed classification method that we explain next.

4. Proposed feature extraction method

In this paper we propose a new method for feature selection that, based on the probabilistic topic model, finds the pairs among all the $|\mathcal{T}_p|$ that are the most discriminative. The feature extraction module is represented in Fig. 3. The input of the system is the set of documents $\Omega_r = (\mathbf{d}_1, \ldots, \mathbf{d}_{|\Omega_r|})$ and the output is a vector of weighted word pairs $\mathbf{g} = \{b_1, \ldots, b_{|\mathcal{T}_{sp}|}\}$, where $\mathcal{T}_{sp}$ is the number of pairs and b_n is the weight associated to each pair (feature) $t_n = (v_i, v_j)$. Therefore, the method works on the initial matrix $\mathcal{T} \times \Omega_r$, where the features are represented as single words. An example of graph structure is showed in Fig. 2.

The graph is made of several clusters, each containing a set of words v_s related to a *keyword* (r_i), a special word which represents the centroid of the cluster. How keywords are selected will be clear further. The weight ρ_{is} can measure how a word is related to a keyword and can be expressed as a probability: $\rho_{is} = P(r_i - v_s)$. The resulting structure is a subgraph rooted on r_i. Moreover, *keywords* can be linked together building a centroids subgraph. The weight

[4] Stemming has sometimes been reported to hurt effectiveness, the recent tendency is to adopt it, as it reduces both the dimensionality of the feature space and the stochastic dependence between terms.

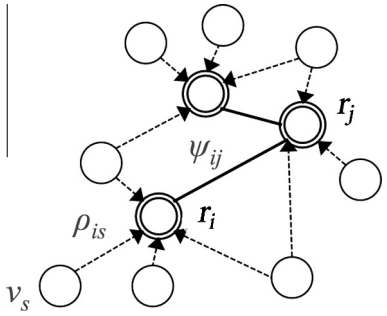

Fig. 2. Graphical representation of the vector of features.

ψ_{ij} can be considered as the degree of correlation between two keywords and can also be expressed as a probability: $\psi_{ij} = P(r_i, r_j)$.

Considering that each keyword is a special word, we can say that the graph contains directed and undirected pairs of features that are all lexically denoted as words. For this reason, the graph can be used to select the most important pairs from the space $\mathcal{T}_p$ in order to obtain a new space $\mathcal{T}_{sp}$, where $|\mathcal{T}_{sp}| \ll |\mathcal{T}_p|$. In this way, the term extraction procedure is obtained by firstly computing all the semantic relatednesses between words and keywords, that is ρ_{is} and ψ_{ij}, and secondly selecting the right subset of pairs from all the possible ones.

Before explaining in detail the learning procedure of a graph, we would like to highlight some aspects of this representation and how we build the graph based classifier.

4.1. Definition of the graph based classifier

As we have seen before, the graph ($\mathbf{g}$) learned from the training set Ω_r can be represented as a vector of features $t_n = (v_i, v_j)$ in the $\mathcal{T}_{sp}$ space. Features can be *word/keyword* or *keyword/keyword* pairs.

By following this approach, also each document of a corpus can be represented in terms of pairs:

$$\mathbf{d}_j = (w_{1j}, \ldots, w_{|\mathcal{T}_{sp}|j}),$$

where w_{nj} is such that $0 \leqslant w_{nj} \leqslant 1$ and represents how much term $t_n = (v_i, v_j)$ contributes to a semantics of document $\mathbf{d}_j$. The weight is calculated thanks to the tf-idf model applied to the pairs represented through t_n:

$$w_{nj} = \frac{\text{tf}-\text{idf}(t_n, \mathbf{d}_j)}{\sqrt{\sum_{n=1}^{|\mathcal{T}_{sp}|} (\text{tf}-\text{idf}(t_n, \mathbf{d}_j))^2}} \tag{3}$$

If we learn a graph $\mathbf{g}_i$ from documents that are labeled as $\mathbf{c}_i$, then $\mathbf{g}_i$ is representative of the training set itself or better is the expert $\hat{\phi}_i$ for the category $\mathbf{c}_i$. Using the expert we can perform the classification by using a linear method that measures the similarity between the expert $\hat{\phi}_i$ and each document $\mathbf{d}_j$ $\forall j$, i, represented in the space $\mathcal{T}_{sp}$. We have considered as a measure of similarity the cosine similarity between vectors in a $\mathcal{T}_{sp}$ space and thus obtaining a ranking classifier $\forall i$:

$$CSV_i(\mathbf{d}_j) = \frac{\sum_{n=1}^{|\mathcal{T}_{sp}|} b_{ni} \cdot w_{nj}}{\sqrt{\sum_{n=1}^{|\mathcal{T}_{sp}|} b_{ni}^2} \cdot \sqrt{\sum_{n=1}^{|\mathcal{T}_{sp}|} w_{nj}^2}} \tag{4}$$

Such a ranking classifier for the category $\mathbf{c}_i \in \mathcal{C}$ consists in the definition of a function, the cosine similarity, that, given a document $\mathbf{d}_j$, returns a *categorization status value* ($CSV_i(\mathbf{d}_j)$) for it, i.e. a real number between 0 and 1 that represents the evidence for the fact that $d_j \in \mathbf{c}_i$, or in other words it is a measure of vector closeness in $|\mathcal{T}_{sp}|$-dimensional space.

Following this criterion each document is then ranked according to its CSV_i value, and so the system works as a document-ranking text classifier, namely a "soft" decision based classifier. As we have discussed in previous sections we need a binary classifier, also known as a "hard" classifier, that is capable of assigning to each document a value T or F to measure the vector closeness.

A way to turn a soft classifier into a hard one is to define a threshold γ_i such that $CSV_i(\mathbf{d}_j) \geqslant \gamma_i$ is interpreted as T while $CSV_i(\mathbf{d}_j) \leqslant \gamma_i$ is interpreted as F. We have adopted an experimental method, that is the *CSV thresholding* (Sebastiani, 2002), which consists in testing different values for γ_i on a subset of the training set (the *validation* set) and choosing the value which maximizes effectiveness. Next we show how such thresholds have been experimentally set.

5. Graph building

A graph $\mathbf{g}$ is learned from a corpus of documents as a result of two important phases: the *Relations Learning* stage, where graph relation weights are learned by computing probabilities between word pairs (see Fig. 3); the *Structure Learning* stage, which specify the shape, namely the structure, of the graph. This stage is achieved by performing an iterative procedure which, given the number of keywords H and the desired max number of pairs as constraints, chooses the best parameter settings τ and $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_H)$ defined as follows:

1. τ: the threshold that establishes the number of *keyword/keyword* pairs of the graph. A relationship between the keyword v_i and keyword r_j is relevant if $\psi_{ij} \geqslant \tau$.
2. μ_i: the threshold that establishes, for each keyword i, the number of *keyword/word* pairs of the graph. A relationship between the word v_s and the keyword r_i is relevant if $\rho_{is} \geqslant \mu_i$.

5.1. Relations Learning

Since each keyword is lexically represented by a word of the vocabulary, we can write $\rho_{is} = P(r_i|v_s) = P(v_i|v_s)$, and $\psi_{ij} = P(r_i, r_j) = \psi_{ij} = P(r_i, r_j) = P(v_i, v_j)$. Considering that $P(v_i, v_j) = P(v_i|v_j)P(v_j)$, all the relations between words result from the computation of the joint or the conditional probability $\forall i, j \in \{1, \ldots, |\mathcal{T}|\}$ (where $|\mathcal{T}|$ is the size of the vocabulary which contains all the indexed words from the corpus) and $P(v_j)$ $\forall j$. An exact calculation of $P(v_j)$ and an approximation of the joint, or conditional, probability can be obtained through a smoothed version of the generative model introduced in Blei et al. (2003) called Latent Dirichlet Allocation (LDA), which makes use of Gibbs sampling (Griffiths et al., 2007). The original theory introduced in Griffiths et al. (2007) mainly proposes a semantic representation in which documents are represented in terms of a set of probabilistic topics z. Formally, we consider a word u_m of the document $\mathbf{d}_m$ as a random variable on the vocabulary $\mathcal{T}$ and z as a random variable representing a topic between $\{1, \ldots, K\}$. A document $\mathbf{d}_m$ results from generating each of its words. To obtain a word, the model considers three parameters assigned: α,η and the number of topics K. Given these parameters, the model chooses θ_m through $P(\theta|\alpha) \sim Dirichlet(\alpha)$, the topic k through $P(z|\theta_m) \sim Multinomial(\theta_m)$ and $\beta_k \sim Dirichlet(\eta)$. Finally, the distribution of each word given a topic is $P(u_m|z, \beta_z) \sim Multinomial(\beta_z)$. The output obtained by performing Gibbs sampling on a set of documents Ω_r consists of two matrixes:

1. the *words-topics* matrix that contains $|\mathcal{T}| \times K$ elements representing the probability that a word v_i of the vocabulary is assigned to topic k:$P(u = v_i|z = k, \beta_k)$;
2. the *topics-documents* matrix that contains $K \times |\Omega r|$ elements representing the probability that a topic k is assigned to some word token within a document $\mathbf{d}_m$:$P(z = k|\theta_m)$.

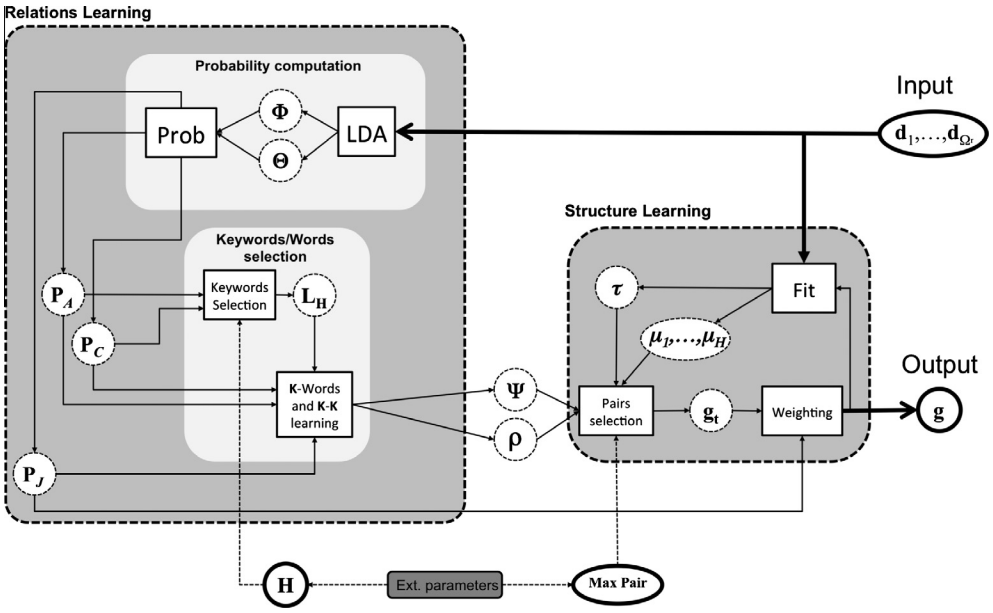

Fig. 3. Proposed feature extraction method. A graph **g** is extracted from a corpus of training documents Ω_r. The quantities $P_J = \psi_{ij}$, $P_C = \rho_{is}$ and $P_A = \eta_s$ are the joint, conditional and a priori probabilities respectively.

The probability distribution of a word within a document $\mathbf{d}_m$ of the corpus can be then obtained as:

$$P(u_m) = \sum_{k=1}^{K} P(u_m|z=k,\beta_k)P(z=k|\theta_m). \tag{5}$$

In the same way, the joint probability between two words u_m and y_m of a document $\mathbf{d}_m$ of the corpus can be obtained by assuming that each pair of words is represented in terms of a set of topics z and then:

$$P(u_m,y_m) = \sum_{k=1}^{K} P(u_m,y_m|z=k,\beta_k)P(z=k|\theta_m) \tag{6}$$

Note that the exact calculation of Eq. (6P) depends on the exact calculation of $P(u_m,y_m{-}z=k,\ \beta_k)$ that cannot be directly obtained through LDA. If we assume that words in a document are conditionally independent given a topic, an approximation for Eq. (6) can be written as:

$$P(u_m,y_m) \simeq \sum_{k=1}^{K} P(u_m|z=k,\beta_k)P(y_m|z=k,\beta_k)P(z=k|\theta_m). \tag{7}$$

Moreover, Eq. (5) gives the probability distribution of a word u_m within a document $\mathbf{d}_m$ of the corpus. To obtain the probability distribution of a word u independently of the document we need to sum over the entire corpus:

$$P(u) = \sum_{m=1}^{M} P(u_m)\delta_m \tag{8}$$

where δ_m is the prior probability for each document $\left(\sum_{m=1}^{|\Omega r|}\delta_m = 1\right)$. In the same way, if we consider the joint probability distribution of two words u and y, we obtain:

$$P(u,y) = \sum_{m=1}^{M} P(u_m,y_v)\delta_m \tag{9}$$

Concluding, once we have $P(u)$ and $P(u,y)$ we can compute $P(v_i) = P(u = v_i)$ and $P(v_i,v_j) = P(u = v_i, y = v_j),\ \forall i,j \in \{1,\ldots,|\mathcal{T}|\}$ and so the relations learning can be totally accomplished.

5.2. Structure learning

Once each ψ_{ij} and ρ_{is} is known $\forall i,j,s$, we need to select the most discriminative pairs of keywords/keywords and keywords/words in order to build the **g** structure. The first step is to select from the words of the indexed corpus a set of keywords $\mathbf{r} = (r_1,\ldots,r_H)$, which will be the nodes of the centroids subgraph. Keywords are meant to be the words whose occurrence is most implied by the occurrence of other words of the corpus, so they can be chosen as follows:

$$r_i = \text{argmax}_{v_i} \prod_{j\neq i} P(v_i|v_j) \tag{10}$$

Since relationships' strengths between keywords can be directly obtained from ψ_{ij}, the centroids subgraph can be easily determined. Note that not all possible relationships between keywords are relevant: the threshold τ can be used as a free parameter for optimization purposes.

As discussed before, several words can be related to each keyword, obtaining H keywords' subgraphs. The threshold set $\boldsymbol{\mu} = (\mu_1,\ldots,\mu_H)$ can be used to select the number of relevant pairs for each keyword's subgraph. Note that a relationship between the word v_s and the keyword r_i is relevant if $\rho_{is} \geqslant \mu_i$, but the value ρ_{is} cannot be directly used to express relationships' strengths between keywords and words. Since ρ_{is} a conditional probability, it is always bigger than ψ_{is} which is a joint probability. Therefore, once pairs for the keyword's subgraph are selected using ρ_{is}, relationships' strength are represented on the graph through ψ_{is}.

Given H and the maximum number of pairs as constraints (i.e. fixed by the user), several structure $\mathbf{g}_t$ can be obtained by varying the parameters $\Lambda_t = (\tau,\mu)_t$. As showed in Fig. 3, an optimization phase is carried out in order to search the set of parameters Λ_t which produces the best graph. This process relies on a scoring function and a searching strategy Bishop (2006) that will be now explained. As we have previously seen, a $\mathbf{g}_t$ is a vector of features $\mathbf{g}_t = \{b_{1t},\ldots,b_{|\mathcal{T}_{sp}|t}\}$ in the space $\mathcal{T}_{sp}$[5] and each document of the

[5] Note that we have a different space of pairs for each set of parameters $\Lambda_t = (\tau,\mu)_t$, therefore the right notation of the space would be $\mathcal{T}_{sp}t$. For the sake of simplicity we omitted the symbol t.

training set Ω_r can be represented as a vector $\mathbf{d}_m = (w_{1m}, \ldots, w_{|\mathcal{T}_{sp}|m})$ in the space $\mathcal{T}_{sp}$. A possible scoring function is the cosine similarity between these two vectors:

$$\mathcal{S}(\mathbf{g}_t, \mathbf{d}_m) = \frac{\sum_{n=1}^{|\mathcal{T}_{sp}|} b_{nt} \cdot w_{nm}}{\sqrt{\sum_{n=1}^{|\mathcal{T}_{sp}|} b_{nt}^2} \cdot \sqrt{\sum_{n=1}^{|\mathcal{T}_{sp}|} w_{nm}^2}} \tag{11}$$

and thus the optimization procedure would consist in searching for the best set of parameters Λ_t such that the cosine similarity is maximized $\forall \mathbf{d}_m$. Therefore, the best $\mathbf{g}_t$ for the set of documents Ω_r is the one that produces the maximum score attainable for each document when used to rank Ω_r documents. Since a score for each document $\mathbf{d}_m$ is obtained, we have:

$$\mathbf{S}_t = \{\mathcal{S}(\mathbf{g}_t, \mathbf{d}_1), \ldots, \mathcal{S}(\mathbf{g}_t, \mathbf{d}_{|\Omega_r|})\},$$

where each score depends on the specific set $\Lambda_t = (\tau, \mu)_t$. To compute the best value of Λwe can maximize the score value for each document, which means that we are looking for the graph which best describes each document of the repository from which it has been learned. It should be noted that such an optimization maximizes at the same time all $-\Omega_r-$ elements of $\mathbf{S}_t$. Alternatively, in order to reduce the number of the objectives being optimized, we can at the same time maximize the mean value of the scores and minimize their standard deviation, which turns a multi-objective problem into a two-objective one. Additionally, the latter problem can be reformulated by means of a linear combination of its objectives, thus obtaining a single objective function, i.e., *Fitness* ($\mathcal{F}$), which depends on Λ_t,

$$\mathcal{F}(\Lambda_t) = E[\mathbf{S}_t] - \sigma[\mathbf{S}_t],$$

where $E$is the mean value of all the elements of $\mathbf{S}_t$ and $\sigma_m$is the standard deviation. By summing up, the parameters learning procedure is represented as follows,

$$\Lambda^* = \text{argmax}_t\{\mathcal{F}(\Lambda_t)\} \tag{12}$$

Since the space of possible solutions could grow exponentially, $|\mathcal{T}_{sp}| \leqslant 300$[6] has been considered. Furthermore, the remaining space of possible solutions has been reduced by applying a clustering method, that is the *K-means* algorithm, to all ψ_{ij} and ρ_{is} values, so that the optimum solution can be exactly obtained after the exploration of the entire space.

This reduction allows us to compute a graph from a repository composed of a few documents in a reasonable time (e.g. for 3 documents it takes about 3 s with a Mac OS X based computer, 2.66 GHz Intel Core i7 CPU and a 8 GB RAM). Otherwise, we would need an algorithm based on a random search procedure in big solution spaces. For instance, Evolutionary Algorithms would be suitable for this purpose, but would provide a slow performance ?? In Fig. 4(a) we can see an example of a graph learned from a set of documents labeled as topic *corn* and in Fig. 4(b) we can see an example of the topic *ship*.

5.3. Extracting a simpler representation from the graph

From the graph we can select different subsets of features so obtaining a simpler representation. Before discussing this in detail, we would recall that $\psi_{ij} = P(v_i, v_j)$ and $\rho_{is} = P(v_i|v_s)$ are computed through the topic model which also computes the probability for each word $\eta_s = P(v_s)$.

We can obtain the simplest representation by selecting from the graph all distinct terms and associating to each of them its weight $\eta_s = P(v_s)$. We name this representation the *List of Terms* ($\mathbf{w}$).

By using the list of terms we can perform a linear classification task considering both vectors of features and documents represented as vectors in the space $\mathcal{T}_s$ and by considering the cosine similarity in such a space.

6. Evaluation

We have considered a classic text classification problem performed on the Reuters-21578 repository. This is a collection of 21,578 newswire articles, originally collected and labeled by Carnegie Group, Inc. and Reuters, Ltd.. The articles are assigned classes from a set of 118 topic categories. A document may be assigned several classes or none, but the commonest case is a single assignment (documents with at least one class received an average of 1.24 classes).

For this task we have used the ModApte split which includes only documents that were viewed and assessed by a human indexer, and comprises 9603 training documents and 3299 test documents. The distribution of documents in classes is very uneven and therefore we have evaluated the system only on documents in the 10 largest classes Christopher et al. (2008)[7].

Note that the graph is different from a simple list of key words because of the presence of two features: the relations between terms and the hierarchical differentiation between simple words and keywords. To demonstrate the discriminative property of such features we have to prove that the results obtained by performing the proposed approach are significantly better than the results obtained by performing the same classification task, through the cosine similarity, when the simple list of weighted words extracted from the graph is used as the vector of features.

In a single label, or binary, classifier we have a training set containing examples that are labeled as $\mathbf{c}_i$ or $\bar{\mathbf{c}}_i$. The learned classifier is capable of assigning a new document to the category $\mathbf{c}_i$ or $\bar{\mathbf{c}}_i$. The graph has been learned only from documents labeled as $\mathbf{c}_i$ (positive examples) and documents belonging to the category $\bar{\mathbf{c}}_i$ have not been used. For this reason, our method is not directly comparable with existing methods. Notwithstanding this, we have compared our approach with linear Support Vector Machines (SVM) learned on the same percentage of the training set but using both positive and negative examples. For SVM we have used a method for term selection based on mutual inference.

As a result, the aim of the evaluation phase is twofold:

1. To demonstrate the discriminative property of the graph compared with a method based only on the words from the graph without relations (named the *List of terms*).
2. To demonstrate that the graph achieves a good performance when a small percentage of the training set is employed for each class. Here a comparison with SVM trained on the same percentage of the training set will be reported.

6.1. Measures

As discussed before, we have considered the *any-of problem* and so we have learned 10 two-class classifiers, one for each class, where the two-class classifier for class c is the classifier for the two classes c and its complement $\bar{c}$. For each of these classifiers, we have used several measures considering TP_i as true positive, TN_i as true negative, FP_i as false positive and FN_i as false negative for the category c_i (Sebastiani, 2002; Christopher et al., 2008):

[6] This number is usually employed in the case of text classifier based on Support Vector Machines.

[7] Note that considering the 10 largest classes means 75% of the training set and 68% of the test set.

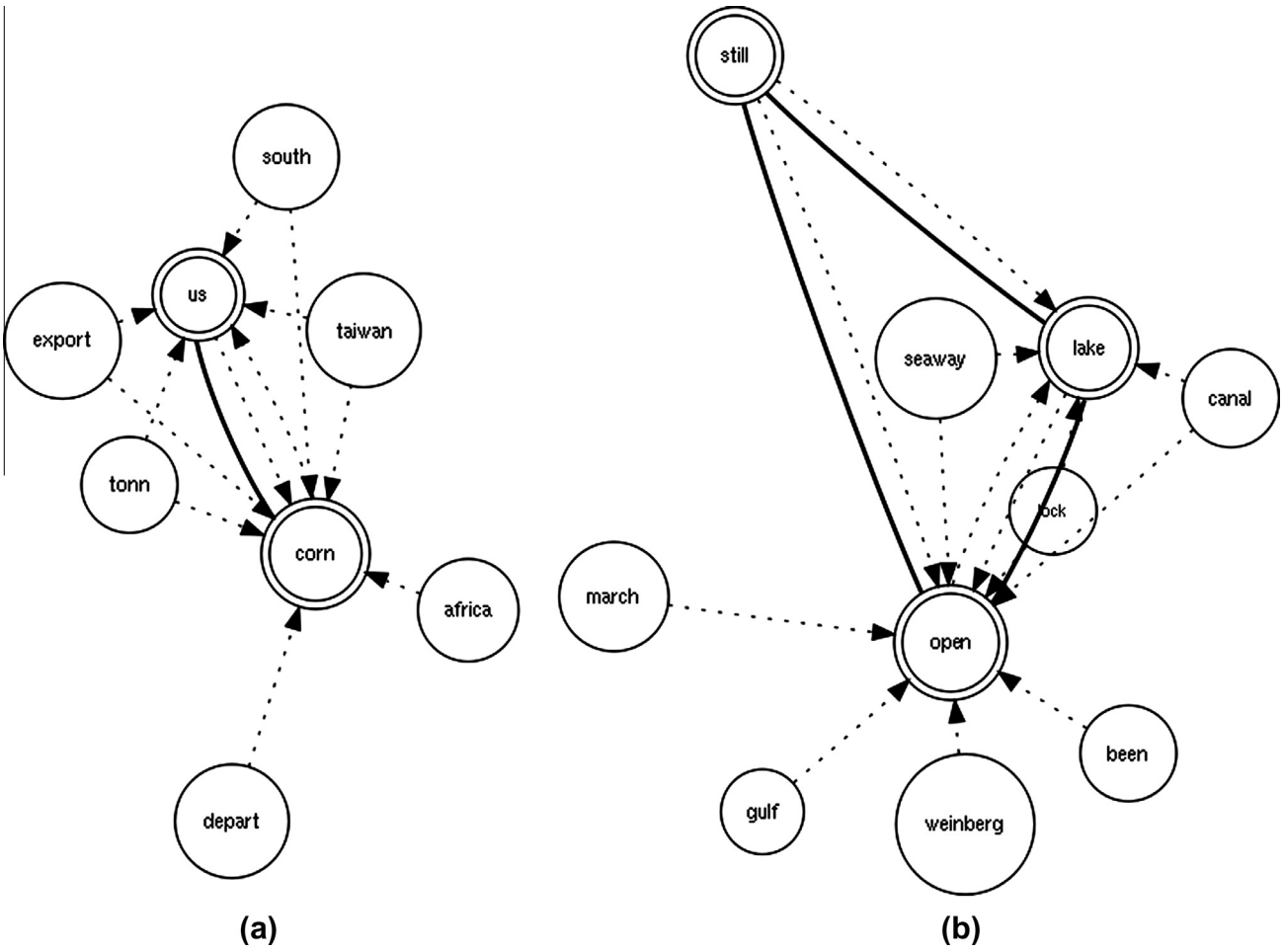

Fig. 4. Part of the *Vector of features*. Solid edges represent undirected relations (ϕ_{ij}) while dotted edges represent directed relations (ρ_{is}). (a) A graph for the topic *corn*. We have 2 keywords (double circles) and 6 words (single circles). (b) A graph for the topic *ship*. We have 3 keywords (double circles) and 8 words (single circles).

- Precision and recall for the category:
  - $P_i = \frac{TP_i}{TP_i+FP_i}$
  - $R_i = \frac{TP_i}{TP_i+FN_i}$
- Micro-average precision and recall:
  - $P_{micro} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} TP_i+FP_i}$
  - $R_{micro} = \frac{\sum_{i=1}^{|C|} TP_i}{\sum_{i=1}^{|C|} TP_i+FN_i}$
- F_1 measure for the category:
  - $F_{1i} = 2 \cdot \frac{P_i \cdot R_i}{P_i+R_i}$
- Micro-average F_1:
  - $F_{1micro} = 2 \cdot \frac{P_{micro} \cdot R_{micro}}{P_{micro}+R_{micro}}$
- Macro-average F_1:
  - $F_{1macro} = \frac{1}{|C|}\sum_{i=1}^{|C|} F_{1i}$

6.2. Experiments

We have set the threshold γ for the categorization status value by evaluating aggregate measures: micro-precision, micro-recall and micro-F1 (see Fig. 5(a)). We have chosen $\gamma = 0.1$ for all the topics.

After the classifier has been set, we have experimented with several dimensions of the reduced training set Υ_r and evaluated the performance through the macro-F_1. In Fig. 5(b) the behavior of the classifier shows a degradation of performance as the dimension of the training set increases. This suggests that the graph of terms becomes less discriminative as the number of labeled examples increases. For this reason, we have considered Υ_r to be about 1% of Ω_r. (See Table 1).

We have randomly selected about 1.4% from each training set (in Table 2 the comparison between the dimension of Υ_r and the original training set Ω_r is reported) and moreover we have performed the selection 100 times in order to make the results independent of the particular documents selection. As a result, for each class we have 100 repositories and from each of them we have calculated 100 graphs by performing the parameters learning described above.

Due to the fact that each optimization procedure leads to a different structure of the graph, we have a different number of pairs for each structure. We have calculated the average number of pairs for each topic and the corresponding average number of terms. Note that the average size of $|\mathcal{T}_{sp}|$ is 116, while the average size of $|\mathcal{T}_s|$ is 33. The overall number of features observed by our method is, independently of the topic, less than the number considered in the case of Support Vector Machines. In fact, we have employed a term selection process obtaining $|\mathcal{T}|_s = 300$.

In Table 2 we have reported the F_1 measure, micro-F_1 and macro-F_1 obtained by the graph **g**, word list **w** and support vector machines (SVM). We have reported the best values and the average

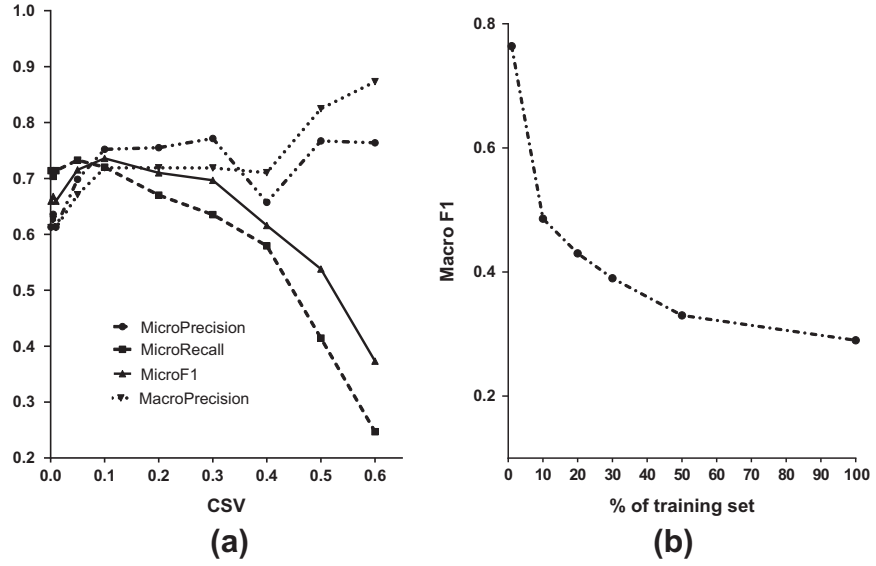

Fig. 5. (a) Tuning of the threshold for γ. (b) Different values of macro-F_1 for different percentages of the training set Υ_r.

Table 1
Average dimension of the reduced training set Υ_r and original dimension of Ω_r.

Topic	Ω_r (KB)	Υ_r (KB)	Perc. (%)
Earn	957	14	1.5
Acq	902	13	1.4
Money-fx	476	7	1.5
Grain	359	5	1.4
Crude	356	5	1.4
Trade	440	6	1.4
Interest	267	4	1.5
Ship	137	2	1.5
Wheat	229	3	1.3
Corn	153	2	1.3
Average	428	6	1.4

Table 2
F_1 measure, F_{1micro} and F_{1macro} for **g**, **w** and SVM. The arrows column shows the increment of **g** performance compared with other methods.

Topic	Graph g		List w		SVM		
	Max	Avg	Max	Avg	Max	Avg	Incr.
Earn	91	76	82	69	95	66	⇓
Acq	63	44	53	38	63	35	⇔
Money-fx	46	30	39	23	37	9	⇑
Grain	66	40	54	35	48	4	⇑
Crude	70	42	60	40	47	10	⇑
Trade	58	42	43	39	27	6	⇑
Interest	50	34	38	24	9	1	⇑
Ship	68	18	59	12	16	1	⇑
Wheat	86	43	72	31	26	2	⇑
Corn	65	23	54	16	10	2	⇑
F_{1micro}	**66**	**39**	46	23	38	14	⇑
F_{1macro}	**74**	**53**	56	33	61	28	⇑

values obtained by performing the classification of all 100 examples of the reduced training set.

It is surprising how the proposed method, even if the training set is smaller than the original one, is capable of classifying in most cases with an accuracy sometimes comparable to and frequently better than Support Vector Machines. Note that the performance of the proposed method is, independently of the topic, better than the word list, so demonstrating that the graph representation possesses better discriminative properties than a simple list of words. Finally, it should be noticed that the good performance showed by the word list based method is due to the fact that the list of words is composed of the terms extracted from the graph demonstrating that the graph could be useful also to select the most discriminative words from the space $\mathcal{T}_s$.

7. Discussion

If we apply the bag of words representation to the reduced training set Υ_r, we obtain a number of features $|\mathcal{T}|$ that is higher than the number of documents: $|\mathcal{T}| \gg |\Upsilon_r|$. In this case, even if we reduce the dimension of Υ_r by selecting the most discriminative features $\mathcal{T}_s$, we may still have $|\mathcal{T}_s| \gg |\Upsilon_r|$ because the number of samples may be too small. As already discussed before in Section 3.1, when $|\mathcal{T}_s| \gg |\Upsilon_r|$ the accuracy of classifiers is poor.

Notwithstanding this, a way to improve the performance of classifiers when $|\mathcal{T}_s| \gg |\Upsilon_r|$ is to employ a method of feature extraction that discovers missing information between features in the original dataset and that maps the discovered information in a new augmented space $\mathcal{T}_p$ where such information can be emphasized. This inspired the proposed vector of features, because the employment of selected pairs of words instead of selected single words improved the correlation between samples of the same class thus helping classifiers to better distinguish across classes.

It is also important to make clear that the graph cannot be considered as a co-occurrence matrix. In fact, the core of the graph is the probability $P(v_i, v_j)$, which we regard as a word association problem, that in the topic model is considered as a problem of prediction: given that a cue is presented, which new words might occur next in that context? It means that the model does not take into account the fact that two words occur in the same document, but that they occur in the same document when a specific topic (and so a context) is assigned to that document (Griffiths et al., 2007).

Furthermore, in the field of statistical learning, a similar structure has been introduced, named the Hierarchical Mixture of Experts (Hastie, Tibshirani, & Friedman, 2009). Such a structure is employed as a method for supervised learning and it is considered as a variant of the well known tree-based methods. The similarity between such a structure and the proposed graph can be obtained by considering the "experts" as "keywords".

Notwithstanding this, the graph is not a tree structure, and more importantly is not rigid but is dynamically built depending on the optimization stage. Moreover, the Hierarchical Mixture of Experts does not consider relations between experts which is, on the other hand, largely employed in the proposed graph. Nevertheless, we will explore further connections between the two methods in future works.

8. Conclusions and future works

The proposed structure implements a document-ranking text classifier, which is able to make a soft decision: it draws up a ranking of documents that requires the choice of an appropriate threshold (Categorization Status Value) in order to obtain a binary classification. This threshold was chosen by evaluating performance on a *validation set* in terms of micro-precision, micro-recall and micro-F1. The dataset Reuters-21578, consisting of about 21 thousand newspaper articles, has been used; in particular, evaluation was performed on the ModApte split (10 categories), which includes only documents classified manually by humans. The experiment was carried out by selecting the 1% randomly in the training set for each category and this selection was made 100 times so that the results are not biased by the specific subset. The performance, evaluated by calculating the F1 measure (harmonic mean of precision and recall), was compared with the Support Vector Machines, in the literature referred as the state of the art in the classification of such a dataset. The results show that when the training set is reduced to 1%, the performance of the graph based classifier are on average higher than those of SVM. This approach can also be applied to Information Retrieval problems (Clarizia, Colace, De Santo, Greco, & Napoletano, 2011), where there is a great interest in discovering solutions that match user information needs and help the decision making process (Rahat Iqbal, 2012; Adam Grzywaczewski, 2012). Note that most text classifiers are trained using both positive and negative examples. An interesting future work could be the use of two graph structures (positive and negative) to observe if there is any improvement in classification performance. Since our graph-based classifier is a document ranking classifier, the combined use of two structures requires a proper choice of the categorization status value. For example, given the document rankings for each graph (positive and negative), scores of documents appearing in both cases, could be properly combined to refine results.

References

Adam Grzywaczewski, R. I. (2012). Task-specific information retrieval systems for software engineers. *Journal of Computer and System Sciences, 78–4*, 1204–1218.

Aggarwal, C., & Zhai, C. (2012). A survey of text classification algorithms. In C. C. Aggarwal & C. Zhai (Eds.), *Mining Text Data* (pp. 163–222). US: Springer.

Berkhin, P. (2006). A survey of clustering data mining techniques. In J. Kogan, C. Nicholas, & M. Teboulle (Eds.), *Grouping multidimensional data* (pp. 25–71). Berlin Heidelberg: Springer.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. New York, NY, USA: Oxford University Press, Inc..

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research, 3*.

Blum, A. L., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence, 97*, 245–271.

Christopher, P. R., SchŸtze, H., & Manning, D. (2008). *Introduction to information retrieval*. Cambridge University.

Clarizia, F., Colace, F., De Santo, M., Greco, L., & Napoletano, P. (2011). A new text classification technique using small training sets. In *Proceedings of 11th international conference on intelligent systems design and applications* (pp. 1038–1043). IEEE Computer Society. ISBN:978-1-4577-1676-8.

Clarizia, F., Colace, F., De Santo, M., Greco, L., & Napoletano, P. (2011). Mixed graph of terms for query expansion. In *Proceedings of 11th international conference on intelligent systems design and applications* (pp. 581–586). IEEE Computer Society. ISBN: 978-1-4577-1676-8.

Colace, F., Santo, M., Greco, L., & Napoletano, P. (2013). Improving text retrieval accuracy by using a minimal relevance feedback. In A. Fred, J. Dietz, K. Liu, & J. Filipe (Eds.), *Knowledge discovery. Knowledge engineering and knowledge management, communications in computer and information science* (Vol. 348, pp. 126–140). Berlin Heidelberg: Springer.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*, 273–297.

Fodor, I. (2002). *A survey of dimension reduction techniques, technical report*.

Griffiths, T. L., Steyvers, M., & Tenenbaum, J. B. (2007). Topics in semantic representation. *Psychological Review, 114*, 211–244.

Hai Dong, E. C., & Hussain, Farookh Khadeer (2011). A framework for discovering and classifying ubiquitous services in digital health ecosystems. *Journal of Computer and System Sciences, 78–4*, 687–704.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*. Springer.

Karavasilis, I., Zafiropoulos, K., & Vrana, V. (2010). A model for investigating e-governance adoption using tam and doi. *International Journal of Knowledge Society Research, 3*, 71–86.

Ko, Y., & Seo, J. (2009). Text classification from unlabeled documents with bootstrapping and feature projection techniques. *Information Processing Management, 45*, 70–83.

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research, 5*, 361–397.

Liu, B. (2006). *Web data mining: Exploring hyperlinks, contents, and usage data (data-centric systems and applications)*. New York, Inc., Secaucus, NJ, USA: Springer-Verlag.

Liu, B., Hsu, W., Ma, Y., & Chen, S. (1999). Mining interesting knowledge using DM-II. In *Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining, KDD '99* (pp. 430–434). New York, NY, USA: ACM.

Magdalini Eirinaki, J. S., & Pisal, Shamita (2012). Feature-based opinion mining and ranking. *Journal of Computer and System Sciences, 78–4*, 1175–1184.

McCallum, A. K. (1996). *Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering*. <http://www.cs.cmu.edu/?mccallum/bow>.

McCallum, A. K. (2002). *Mallet: A machine learning for language toolkit*. <http://mallet.cs.umass.edu>.

McCallum, A., & Nigam, K. (1998). *A comparison of event models for naive Bayes text classification*.

McCallum, A., Nigam, K., Rennie, J., & Seymore, K. (1999). A machine learning approach to building domain-specific search engines. *Proceedings of the 16th international joint conference on Artificial intelligence* (Vol. 2, pp. 662–667). Morgan Kaufman.

Napoletano, P., Colace, F., De Santo, M., & Greco, L. (2012). Text classification using a graph of terms. In *2012 Sixth international conference on complex, intelligent and software intensive systems (CISIS)* (pp. 1030 –1035).

Ng, A.Y., Jordan, M. I. (2002). *On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes*.

Noam, S. Naftali, T. (2001). The power of word clusters for text classification. In *23rd European colloquium on information retrieval research*.

Palus, S., Bródka, P., & Kazienko, P. (2011). Evaluation of organization structure based on email interactions. *International Journal of Knowledge Society Research, 2*, 1–13.

Quinlan, J. (1986). Induction of decision trees. *Machine Learning, 1*, 81–106.

Rahat Iqbal, N. S. (2012). Information retrieval, decision making process and user needs. *Journal of Computer and System Sciences, 78–4*, 1158–1159.

Salton, G., & McGill, M. J. (1983). *Introduction to modern information retrieval*. McGraw-Hill.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surverys, 34*, 1–47.

Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: An overview. *International Journal Data Warehousing and Mining*, 1–13.

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval, SIGIR '99* (pp. 42–49). New York, NY, USA: ACM.