Learning to Classify Text Using a Few Labeled Examples

Conference Paper *in* Communications in Computer and Information Science · September 2013

DOI: 10.1007/978-3-642-37186-8_13

CITATIONS
0

READS
95

4 authors:

F. Colace
Università degli Studi di Salerno
206 PUBLICATIONS **2,308** CITATIONS
SEE PROFILE

Massimo De Santo
Università degli Studi di Salerno
218 PUBLICATIONS **2,303** CITATIONS
SEE PROFILE

Luca Greco
Università degli Studi di Salerno
53 PUBLICATIONS **986** CITATIONS
SEE PROFILE

Paolo Napoletano
Università degli Studi di Milano-Bicocca
154 PUBLICATIONS **4,107** CITATIONS
SEE PROFILE

Learning to Classify Text Using a Few Labeled Examples

Francesco Colace[1], Massimo De Santo[1], Luca Greco[1], and Paolo Napoletano[2,⋆]

[1] Department of Electronic Engineering and Computer Engineering,
University of Salerno, 84084 Fisciano, Italy
{fcolace,desanto,lgreco}@unisa.it
[2] DISCo (Department of Informatics, Systems and Communication)
University of Milan, Bicocca Viale Sarca 336
20126 Milan, Italy
napoletano@disco.unimib.it

Abstract. It is well known that supervised text classification methods need to learn from many labeled examples to achieve a high accuracy. However, in a real context, sufficient labeled examples are not always available. In this paper we demonstrate that a way to obtain a high accuracy, when the number of labeled examples is low, is to consider structured features instead of list of weighted words as observed features. The proposed vector of features considers a hierarchical structure, named a mixed Graph of Terms, composed of a directed and an undirected sub-graph of words, that can be automatically constructed from a set of documents through the probabilistic Topic Model.

Keywords: Text classification, Term extraction, Probabilistic topic model.

1 Introduction

The problem of supervised *text classification* has been extensively discussed in literature where metrics and measures of performance have been reported [4], [12], [9]. All the existing techniques have been demonstrated to achieve a high accuracy when employed in supervised classification tasks of large datasets.

Nevertheless, it has been found that only 100 documents can be hand-labeled in 90 minutes and in this case the accuracy of classifiers (amongst which we find Support Vector Machine based methods), learned from this reduced training set, could be around 30% [8].

This makes, most times, a classifier unfeasible in a real context. In fact, most users of a practical system do not want to carry out labeling tasks for a long time only to obtain a higher level of accuracy. They obviously prefer algorithms that have a high accuracy, but do not require a large amount of manual labeling tasks [10][8]. As a consequence, we can affirm that, in several application fields we need algorithms to be fast and with a good performance.

Although each existing method has its own properties, there is a common denominator: the "bag of words" assumption to create term weights.

The "bags of words" assumption claims that a document can be considered as a *vector of features* where each element in the vector indicates the presence (or absence)

⋆ Corresponding author.

A. Fred et al. (Eds.): IC3K 2011, CCIS 348, pp. 200–214, 2013.

of a word, so that the information on the position of that word within the document is completely lost [4].

It is well known that the main purpose of text mining techniques is to identify common patterns through the observation of such *vectors of features* and then to use such patterns to make predictions. Unfortunately, the accuracy of classification methods based on the "bags of words" decreases as the number of labeled examples decreases. In this case classifiers identify common patterns that are insufficiently discriminative because, due to the inherent ambiguity of language (polysemy etc.), vectors of weighted words are not capable of discriminating between documents.

The ambiguity, in fact, can be reduced if we give more importance to words that convey concepts and that contribute to specify a topic, and if we assign less importance to those words that contribute to specify concepts and that, due to the fact that they can be more plausibly shared between concepts, can increase the ambiguity.

This leads to a hierarchical structure that we call a mixed *Graph of Terms* and that can be automatically extracted from a set of documents $\mathcal{D}$ using a global method for term extraction based on Latent Dirichlet Allocation [2] implemented as Probabilistic Topic Model [6].

Here we propose a linear single label supervised classifier that is capable, based on a *vector of features* represented through a mixed *Graph of Terms*, of achieving a better performance, in terms of accuracy, than existing methods when the size of the training set is about 1.4% of the original and composed of only positive examples.

To confirm the discriminative property of the graph we have evaluated the performance through a comparison between our methodology and a term selection methodology which considers the *vector of features* formed of only the list of concepts and words composing the graph and so where relations have not been considered. We have also compared our method with linear Support Vector Machines. The 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 Problem Definition

Following the definition introduced in [12], 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 $\mathcal{C} = \{\mathbf{c}_1, ..., \mathbf{c}_{|\mathcal{C}|}\}$ is a predefined set of *categories* and $\mathcal{D}$ is a set of *documents*.

If $\Phi(\mathbf{d}_m, \mathbf{c}_i) = T$, then $\mathbf{d}_m$ is called a positive example (or a member) of $\mathbf{c}_i$, while if $\Phi(\mathbf{d}_m, \mathbf{c}_i) = F$ 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 these cases, the classification must be accomplished only on the basis of knowledge extracted from the documents themselves, namely *endogenous knowledge*.

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

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 $\mathbf{d}_m \in \mathcal{D}$ must be assigned either to $\mathbf{c}_i$ or to its complement $\overline{\mathbf{c}}_i$. 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 [12,13].

It means that we consider the classification under $\mathcal{C} = \{\mathbf{c}_1, ..., \mathbf{c}_{|\mathcal{C}|}\}$ 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 $\hat{\Phi} = \{\hat{\phi}_1, \ldots, \hat{\phi}_{|\mathcal{C}|}\}$.

Once the classification problem has been defined we can start the pre-processing of the data through the data preparation and reduction steps.

2.1 Data Preparation

Texts can not be directly interpreted by a classifier and for this reason, an indexing procedure that maps a text $\mathbf{d}_m$ into a compact representation of its content must be uniformly applied to the training and test documents. In the following we consider the case of the training set.

Each document can be represented, following the *Vector Space Model* [4], as a vector of term *weights*

$$\mathbf{d}_m = \{w_{1m}, \ldots, w_{|\mathcal{T}|m}\},$$

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 $0 \leq w_{nm} \leq 1$ represents how much term t_n contributes to a semantics of document $\mathbf{d}_{mm}$.

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_{nm} indicates the presence (or absence) of a word, so that the information on the position of that word within the document is completely lost [4].

To determine the weight w_{nm} of term t_n in a document $\mathbf{d}_m$, the standard tf-idf (*term frequency-inverse document frequency*) function can be used [11], defined as:

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

where $N(t_n, \mathbf{d}_m)$ denotes the number of times t_n occurs in $\mathbf{d}_m$, 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.

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:

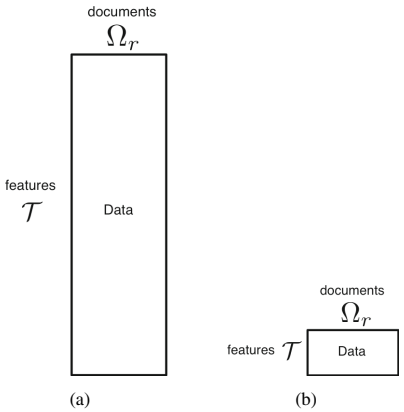


Fig. 1. Features-documents matrix. 1(a) In this case the number of features is much higher than the number of examples ($|\mathcal{T}| \gg |\Omega_r|$). 1(b). In this case $|\mathcal{T}| \ll |\Omega_r|$.

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

In this paper, before indexing, we have performed the removal of function words (i.e. topic-neutral words such as articles, prepositions, conjunctions, etc.) and we have performed the stemming procedure[1] (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.

2.2 Data 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. In this case, the *computational cost* of the learning procedure and *overfitting* of the classifier are very common problems [3].

Moreover, 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 the case of text documents the number of features is usually high and particularly it is usually higher than the number of documents. In Fig. 1(a) we show the case of a training set composed of 100 documents and about 20000 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).

[1] 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.

To deal with these issues, dimension *reduction techniques* are applied as a data preprocessing 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 [3,12,4,5]:

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 transformation* that considers pairs of words instead of single words as basic features thus obtaining a new space $\mathcal{T}_p$ of features. The dimensionality of such a new space is very high, much higher than $|\mathcal{T}|$, in fact: $|\mathcal{T}_p| \propto |\mathcal{T}|^2$. 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* [2] implemented as the *Probabilistic Topic Model* [6] and this is the core of the proposed classification method that we explain next.

3 Proposed Feature Selection 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 method works on the initial data representation, that is the matrix $\mathcal{T} \times \Omega_r$, where the features are the single words, and extracts a new representation, named the mixed *Graph of Terms*, that consists of related pairs of words. The graph contains two kinds of relations between words, directed and undirected, and for this reason it is called *mixed*.

In the graph we can find several clusters of words and each cluster contains a set of words v_s that specify, through a directed weighted edge, a special word, that we have named the *concept*, r_i, that is the centroid of such a cluster. The weight ρ_{is} can measure how far a word is related to a concept, or how much we need such a word to specify

that concept, and it can be considered as a probability: $\rho_{is} = P(r_i|v_s)$. The resulting structure is a subgraph rooted on r_i (see fig. 2(a)).

Moreover, special words, namely *concepts*, can be linked together through undirected weighted edges, so forming a subgraph of pairs of centroids. The weight ψ_{ij} can be considered as the degree of semantic correlation between two concepts and it can be considered as a probability: $\psi_{ij} = P(r_i, r_j)$ (see fig. 2(a)).

Considering that each concept 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 reduced space $\mathcal{T}_{sp}$.

Given the training set Ω_r of documents, the proposed method, through a learning procedure, selects a subset of pairs obtaining a number of pairs $|\mathcal{T}_{sp}| \ll |\mathcal{T}_p|$. In this way, the term extraction procedure is obtained by firstly computing all the semantic relatednesses between words and concepts, 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.

3.1 Graph and Document Representation in the Space $\mathcal{T}_{sp}$

A graph $\mathbf{g}$ can be viewed, following the *Vector Space Model* [4], as a vector of features t_n:

$$\mathbf{g} = \{b_1, \ldots, b_{|\mathcal{T}_{sp}|}\},$$

where $|\mathcal{T}_{sp}|$ represents the number of pairs and each feature $t_n = (v_i, v_j)$ can be a *word/concept* or *concept/concept* pair. The weight b_n is named *boost* factor and is equal to ψ_{ij} for both *word/concept* or *concept/concept* pairs.

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

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

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

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

3.2 Classifier Definition in the Space $\mathcal{T}_{sp}$

As we have seen before, the mixed *Graph of Terms* ($\mathbf{g}$) learned from the training set Ω_r can be also represented as a vector of features in the $\mathcal{T}_{sp}$ space. If we learn a graph $\mathbf{g}_i$ from documents that are labeled as $\mathbf{c}_i$, then $\mathbf{g}_i$ it can be considered as representative of such labeled set of documents and considered as the expert $\hat{\phi}_i$ for the category $\mathbf{c}_i$ itself:

$$\mathbf{g}_i = \hat{\phi}_i = \{b_{1i}, \ldots, b_{|\mathcal{T}_sp|i}\}.$$

Using the expert we can perform a classification task by using a linear method that measures the similarity between the expert $\hat{\phi}_i$ and each document $\mathbf{d}_m$ represented in the space $\mathcal{T}_{sp}$.

Here we have considered as a measure of similarity the well known cosine similarity between vectors in a $\mathcal{T}_{sp}$ space and thus obtaining a ranking classifier $\forall i$:

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

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 $\mathbf{d}_m$, returns a *categorization status value* ($CSV_i(\mathbf{d}_m)$) for it, i.e. a number between 0 and 1 that represents the evidence for the fact that $\mathbf{d}_m \in \mathbf{c}_i$, or in other words it is a measure of vector closeness in a $|\mathcal{T}_{sp}|$-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 $CSV_i(\mathbf{d}_m) \geq \gamma_i$ is interpreted as T while $CSV_i(\mathbf{d}_m) \leq \gamma_i$ is interpreted as F. We have adopted an experimental method, that is the CSV *thresholding* [12], 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.

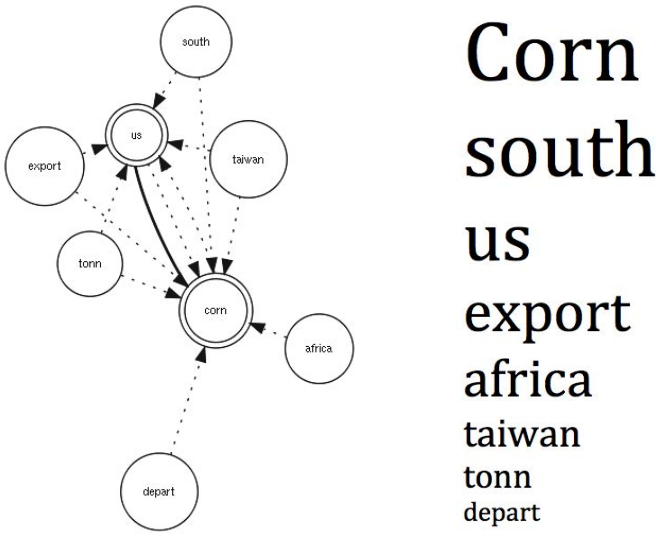
4 Graph Learning

A graph $\mathbf{g}$ is well determined through the learning of the weights, the *Relations Learning* stage, and through the learning of three parameters, the *Structure Learning* stage, that are $\Lambda = (H, \tau, \mu)$ which specify the shape, namely the structure, of the graph. In fact, we have:

1. H: the number of concepts (namely the number of clusters) of the set of documents;
2. μ_i: the threshold that establishes for each concept the number of edges of the directed subgraph, and so the number of *concept/word* pairs of the corpus . An edge between the word s and the concept i can be saved if $\rho_{is} \geq \mu_i$. To simplify the formulation, we assume that $\mu_i = \mu, \forall i$;
3. τ: the threshold that establishes the number of edges of the undirected subgraph, and so the number of *concept/concept* pairs of the corpus. An edge between the concept i and concept j can be saved if $\psi_{ij} \geq \tau$.

4.1 Relations Learning

Due to the fact that each concept is lexically represented by a word of the vocabulary, then we have that $\rho_{is} = P(r_i|v_s) = P(v_i|v_s)$, and $\psi_{ij} = P(r_i, r_j) = P(v_i, v_j)$.



(a) (b)

Fig. 2. Part of the *Vector of features* for the topic *corn*. We have 2 concepts (double circles) and 6 words (single circles). Solid edges represent undirected relations (ψ_{ij}) while dotted edges represent directed relations (ρ_{is}). 2(a) A mixed *Graph of Terms*. 2(b) A *List of Terms*.

Considering that $P(v_i, v_j) = P(v_i|v_j)P(v_j)$, it is necessary, to learn all the relations between words, to compute the joint, or the conditional, probability $\forall i, j \in \{1, \cdots, |\mathcal{T}|\}$ and each $P(v_j)$ $\forall j$.

We show here that the exact calculation of $P(v_j)$ and the approximation of the joint, or conditional, probability can be obtained through a smoothed version of the generative model introduced in [2] called Latent Dirichlet Allocation (LDA), which makes use of Gibbs sampling [6].

The original theory introduced in [6] mainly asserts 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 $\{1, \cdots, K\}$. The probability distribution of a word within a document $\mathbf{d}_m$ of the corpus can be obtained as:

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

The generation of a document $\mathbf{d}_m$ can be obtained considering the generation of each word of the document. 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(α), the topic k through $P(z|\theta_m) \sim$ Multinomial(θ_m) and $\beta_k \sim$ Dirichlet(η). Finally, the distribution of each word given a topic is $P(u_m|z, \beta_z) \sim$ Multinomial(β_z).

As we have already discussed, we have used a smoothed version of Latent Dirichlet Allocation (LDA), which makes use of Gibbs sampling. The results obtained by performing this algorithm on a set of documents Ωr are 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 $\mathbf{d}_m$: $P(z = k|\theta_m)$.

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. 6 depends on the exact calculation of $P(u_m, y_m|z = k, \beta_k)$ that can not be directly obtained through LDA. For this reason, we have introduced an approximation that considers words in a document as conditionally independent given a topic. In this way Eq. 6 can be written as:

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

Note that 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 ($\sum_{m=1}^{|\Omega r|} \delta_m = 1$).

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

4.2 Structure Learning

Given a set of documents, once each ψ_{ij} and ρ_{is} is known $\forall i, j, s$, letting the parameters $\Lambda_t = (H, \tau, \mu)_t$ assume a different set of values, we can observe a different structure of the graph $\mathbf{g}_t$ (here t is representative of different parameter values).

A way to learn the structure of the graph is to use an optimization based algorithm that searches for the best set of parameters Λ_t. In this case we need a scoring function and a searching strategy [1].

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 $\mathcal{T}_{sp}$ and each document of the 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:

$$\mathcal{S}(\mathbf{g}_t, \mathbf{d}_m) = \frac{\sum_{n=1}^{|\mathcal{T}_{sp}|} b_{nt} \cdot w_{nm}}{\sqrt{\sum_{n=1}^{|\mathcal{T}_{sp}|} b_{nt}^2} \cdot \sqrt{\sum_{n=1}^{|\mathcal{T}_{sp}|} w_{nm}^2}} \tag{10}$$

and thus the optimization procedure would consist in searching for the best set of parameters Λ_t such that the cosine similarity is maximized $\forall \mathbf{d}_m$.

By following this approach, the best $\mathbf{g}_t$ for the set of documents Ω_r is the one that produces the maximum score attainable for each of the documents when the same graph is used as a vector of features to measure the similarity of a set containing just those documents which have fed the graph builder.

As a consequence, we obtain a score for each document $\mathbf{d}_m$ and then we have

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

where each score depends on the specific set $\Lambda_t = (H, \tau, \mu)_t$.

To compute the best value of Λ we 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, we can reformulate the latter problem 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,

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

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

$$\Lambda^* = \underset{t}{\operatorname{argmax}}\{\mathcal{F}(\Lambda_t)\}.$$

We will see next how we have performed the searching strategy phase.

Since the space of possible solutions could grow exponentially, we have considered[2] $|\mathcal{T}_{sp}| \leq 300$. Furthermore, we have reduced the remaining space of possible solutions 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.

[2] This number is usually employed in the case of Support Vector Machines.

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 seconds with a Mac OS X based computer, 2.66 GHz Intel Core i7 CPU and a 8GB 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. 2(a) we can see an example of a graph learned from a set of documents labeled as topic *corn*.

4.3 Extracting a Simpler Representation from the Graph

From the mixed *Graph of Terms* we can select different subsets of features so obtaining a simpler representation (see fig. 2(b)). 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* (**w**), see fig. 2(b).

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.

4.4 Consideration on the Method

Here we wish to demonstrate that by using such a graph as a vector of features we are capable of achieving a good performance, in terms of accuracy, even if the size of the training set Υ_r is about 1.4% of the original Ω_r and is composed of only positive examples. We further wish to demonstrate that the performance of our approach is better than existing methods, such as support vector machines, based on feature selection instead of feature extraction.

How can we obtain a good performance when the training set is small? In this case, in fact, the number of documents is low while the number of features (for instance words), that occur in these documents, is higher, $|\mathcal{T}| \gg |\Upsilon r|$. Even if we perform data reduction through a selection method, we could still have $|\mathcal{T}_s| \gg |\Upsilon_r|$. If we follow the theory introduced at the beginning of this paragraph, in this case we have to say that the efficiency and accuracy of data analysis are low.

In this work we wish to demonstrate that a way to improve the performance when $|\mathcal{T}| \gg |\Upsilon_r|$ is to apply 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.

The bags of words representation of a text introduces ambiguity when $|\mathcal{T}| \gg |\Upsilon_r|$ and we argue that the only way to reduce the ambiguity is to introduce another dimension of observation where it is possible to distinguish which word conveys which meaning. By using different pairs of words in the graph we are able to give more importance to words that convey concepts and that contribute to specify a topic and to assign less importance to those words that contribute to specify concepts and that, due

to the fact that they can be more plausibly shared between concepts, can increase the ambiguity.

It is also important to make clear that the mixed *Graph of Terms* can not 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 [6].

Furthermore, in the field of statistical learning, a similar structure has been introduced, named the Hierarchical Mixture of Experts [7]. 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 "concepts".

Notwithstanding this, the mixed Graph of terms 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 mixed Graph of Terms. Nevertheless, we will explore further connections between the two methods in future works.

5 Evaluation

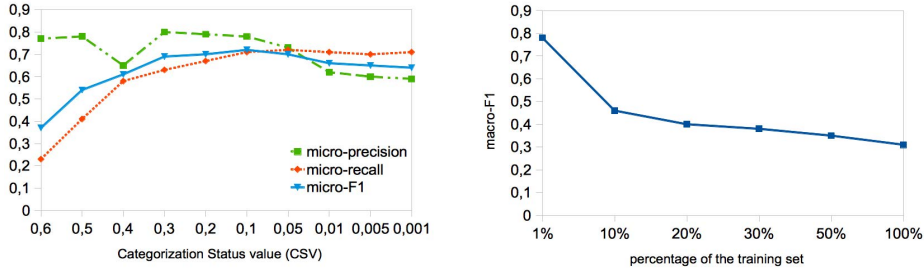
We have considered a classic text classification problem performed on the Reuters-21578 repository. This is a collection of 21,578 newswire articles, originally collected and labeled by Carnegie Group, Inc. and Reuters, Ltd.. The articles are assigned classes from a set of 118 topic categories. A document may be assigned to 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 9,603 training documents and 3,299 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 [4][3].

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 concepts. 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 usually have a training set containing examples that are labeled as $\mathbf{c}_i$ or $\overline{\mathbf{c}}_i$. The learned classifier is capable of assigning a new document to the category $\mathbf{c}_i$ or $\overline{\mathbf{c}}_i$. The graph has been learned only from documents

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



(a) (b)

Fig. 3. 3(a) Tuning of the threshold for γ. 3(b) Different values of macro-F_1 for different percentages of the training set Υ_r.

labeled as $\mathbf{c}_i$ (positive examples) and documents belonging to the category $\overline{\mathbf{c}}_i$ 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 Words List);
2. To demonstrate that the graph achieves a good performance when 1.4% of the training set is employed for each class. Here we report a comparison with SVM trained on the same percentage of the training set.

5.1 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 $\mathbf{c}_i$ is the classifier for the two classes $\mathbf{c}$ and its complement $\overline{\mathbf{c}}_i$. 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 $\mathbf{c}_i$ ([12,4]):

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

Table 1. Average dimension of the reduced training set Υ_r and original dimension of Ω_r. F_1 measure, F_{1micro} and F_{1macro} for the graph ($\mathbf{g}$), word list ($\mathbf{w}$) and support vector machines (SVM). The arrows column shows the increment of $\mathbf{g}$ performance compared with other methods.

Topic	Ω_r(KB)	Υ_r(KB)	**g@max**	**g@av**	**w@max**	**w@av**	**SVM@max**	**SVM@av**	
earn	957	14	91	76	82	69	95	66	$\Downarrow$
acq	902	13	63	44	53	38	63	35	$\Leftrightarrow$
money-fx	476	7	46	30	39	23	37	09	$\Uparrow$
grain	359	5	66	40	54	35	48	04	$\Uparrow$
crude	356	5	70	42	60	40	47	10	$\Uparrow$
trade	440	6	58	42	43	39	27	06	$\Uparrow$
interest	267	4	50	34	38	24	09	01	$\Uparrow$
ship	137	2	68	18	59	12	16	01	$\Uparrow$
wheat	229	3	86	43	72	31	26	02	$\Uparrow$
corn	153	2	65	23	54	16	10	02	$\Uparrow$
		F_{1micro}	**66**	**39**	**46**	**23**	**38**	**14**	$\Uparrow$
		F_{1macro}	**74**	**53**	**56**	**33**	**61**	**28**	$\Uparrow$

5.2 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. 3(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. 3(b) the behavior of the classifier shows a degradation of performance as the dimension of the training set increases. This suggests that the mixed graph of terms becomes less discriminative as the number of labeled examples increases. For this reason, we have considered Υ_r to be about 1.4% of Ω_r.

We have randomly selected the 1.4% from each training set (in table 1 is reported the comparison between the dimension of Υ_r and the original training set Ω_r) and moreover we have performed the selection 100 times in order to make the results independent of the particular document selection. As a result, 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 where we have employed a term selection process obtaining $|\mathcal{T}|_s = 300$.

In table 1 we have reported the F_1 measure, micro-F_1 and macro-F_1 obtained by the graph $\mathbf{g}$, word list $\mathbf{w}$ and support vector machines (SVM). We have reported the

best values and the average 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 mostly 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 shown 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 $\mathcal{T}_s$.

6 Conclusions

In this work we have demonstrated that a *term extraction* procedure based on a mixed *Graph of Terms* representation is capable of achieving a better performance than a method based on a simple *term selection*, obtained considering only the words composing the graph, and a linear version of SVM. Moreover, we have demonstrated that the overall performance of the method is good even when only 1.4% of the training set has been employed.

References

1. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
3. Blum, A.L., Langley, P.: Selection of relevant features and examples in machine learning. Artificial Intelligence 97, 245–271 (1997)
4. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University (2008)
5. Fodor, I.: A survey of dimension reduction techniques. Tech. rep. (2002)
6. Griffiths, T.L., Steyvers, M., Tenenbaum, J.B.: Topics in semantic representation. Psychological Review 114(2), 211–244 (2007)
7. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer (2009)
8. Ko, Y., Seo, J.: Text classification from unlabeled documents with bootstrapping and feature projection techniques. Inf. Process. Manage. 45, 70–83 (2009)
9. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: Rcv1: A new benchmark collection for text categorization research. J. Mach. Learn. Res. 5, 361–397 (2004)
10. McCallum, A., Nigam, K., Rennie, J., Seymore, K.: A machine learning approach to building domain-specific search engines. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence, vol. 2, pp. 662–667. Morgan Kaufmann (1999)
11. Salton, G., McGill, M.J.: Introduction to modern information retrieval. McGraw-Hill (1983)
12. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. 34, 1–47 (2002)
13. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. Int. J. Data Warehousing and Mining 2007, 1–13 (2007)