Information Systems 36 (2011) 843–858

Contents lists available at ScienceDirect

Information Systems

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

Word co-occurrence features for text classification

Fábio Figueiredo [a,b,*], Leonardo Rocha [c], Thierson Couto [d], Thiago Salles [b], Marcos André Gonçalves [b], Wagner Meira Jr. [b]

[a] EconoInfo Research, Belo Horizonte, Brazil
[b] Universidade Federal de Minas Gerais, Computer Science Department, Belo Horizonte, Brazil
[c] Universidade Federal de São João Del Rei, Computer Science Department, São João Del Rei, Brazil
[d] Universidade Federal de Goiás, Institute of Informatics, Goiânia, Brazil

ARTICLE INFO

Article history:
Received 23 September 2008
Received in revised form
12 February 2011
Accepted 24 February 2011
Recommended by: R. Baeza-Yates
Available online 9 March 2011

Keywords:
Classification
Text mining
Feature extraction

ABSTRACT

In this article we propose a data treatment strategy to generate new discriminative features, called compound-features (or c-features), for the sake of text classification. These c-features are composed by terms that co-occur in documents without any restrictions on order or distance between terms within a document. This strategy precedes the classification task, in order to enhance documents with discriminative c-features. The idea is that, when c-features are used in conjunction with single-features, the ambiguity and noise inherent to their bag-of-words representation are reduced. We use c-features composed of two terms in order to make their usage computationally feasible while improving the classifier effectiveness. We test this approach with several classification algorithms and single-label multi-class text collections. Experimental results demonstrated gains in almost all evaluated scenarios, from the simplest algorithms such as *k*NN (13% gain in micro-average F_1 in the 20 Newsgroups collection) to the most complex one, the state-of-the-art SVM (10% gain in macro-average F_1 in the collection OHSUMED).

1. Introduction

The widespread use of the Internet for distributing all sorts of information resulted in an ever increasing amount of data being stored and accessible through the WWW, among other applications. These data are frequently organized as textual documents and have been the main target of search engines and other retrieval tools, which perform tasks such as searching and filtering. One popular and traditional technique associated with such tasks is automatic text classification (ATC) [1], that is, assigning documents to one or more pre-defined categories. Text classification allows, for instance, easy and semantically meaningful grouping of documents, helping both users and information retrieval tools to locate them. Currently, text classification is the basis of several popular and relevant applications such as spam filtering [2,3], adult content detection [4–6], and automatic labeling of digital library documents [7,8].

Similarly to other classification techniques, ATC is usually implemented through a supervised learning process, that is, there is a training set of already classified samples, and these samples are used for creating a classifier (learned model). Once ready, the classifier is used for determining the best category for a non-classified document [9]. The process of constructing a classifier may be seen as determining a set of criteria (usually the occurrence of terms/words or other features such as citations) that partition the training documents into groups that are as homogeneous as possible with respect to their categories.

* Corresponding author at: Universidade Federal de Minas Gerais, Computer Science Department, Belo Horizonte, Brazil.
Tel.: +55 31 3037 7070; fax: +55 31 3409 5858.
E-mail address: fab@dcc.ufmg.br (F. Figueiredo).

0306-4379/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2011.02.002

One common challenge for obtaining good models is that the training set may not have enough discriminative features, providing ambiguous information. There are also the limitations of the classifier in terms of the complexity of the patterns that it is able to represent, which should account for the dependences among features and their varying discriminative power. We illustrate these issues through three popular ATC techniques: *k*NN, Naïve-Bayes, and SVM. These techniques employ a single classification function that considers all features, generating a ranking of the possible categories to be assigned to a novel document. Ambiguous features affect the accuracy of the classification functions because they lower the average discriminative power of their learned models. Thus, the lack of discriminative features may lead to poor accuracy. In summary, one major issue is how to minimize the impact of non-discriminative features while building models.

In this paper we propose and evaluate a strategy for feature extraction, which aims to augment in advance both training and test documents with novel features that will ease the classifier task. Intuitively, our proposal augments documents by adding compound-features (hereafter called c-features) that represent the co-occurrence of two or more terms and provide good discriminative ability. To illustrate our argument, let us assume we have a medical digital library containing documents about various medical subjects. This is the case of the OHSUMED collection, one of the datasets used in this work and detailed later. In the OHSUMED collection, the term *pain* may appear in several contexts, such as *nervous system diseases*, *musculoskeletal diseases* and *pathological conditions, signs and symptoms*. In this case, the usage of pairs of terms (e.g., {pain,facial}, {pain,hips}, and {pain,postoperative}) may help determining the document category. Notice that, in all cases, the terms alone do not mean much, while their usage together with *pain* brings much more information. At the same time, some other pairs of terms that do co-occur in the documents, such as {pain,reach} and {pain,beliefs}, might not add relevant information for the classification task, and should be filtered out for sake of discarding irrelevant data or noise. These examples illustrate the gains that can be provided by feature extraction.[1]

It is interesting to understand how feature extraction may help to improve different classifier models. By adding good discriminative features we may, for instance, (i) better approximate a test document to neighbor documents belonging to its true class, in the case of *k*NN classifier, (ii) increase the probability that a test document belongs to its true class, in the case of Naïve-Bayes classifier and (iii) confirm that some compound-features were considered more discriminative than their individual terms, according to weights assigned to them by SVM. Thus, one major issue for improving the classifier is how to obtain a set of features that are discriminative and help classifying text. A second major issue is associated with the computational cost of feature extraction, since the number of possible feature combinations for a collection containing T terms is 2^T.

It is also important to compare how feature extraction helps to improve the classifier in comparison to feature selection. In fact, they are complementary. Feature selection ranks the features according to their discriminative power, discarding both noisy and irrelevant features [10,11]. On the other hand, feature extraction may add new relevant features to produce better classifiers. Our strategy exploits co-occurrence of terms in the documents of a given class, that is, if a c-feature occurs in the documents of a given class and just in this class, the c-feature has good discriminative value, even when the terms that compose it are not good discriminants. Thus, one immediate effect of feature extraction is that we reduce the impact of ambiguous terms, since we can narrow down the possible meanings of some of the terms when we consider the interaction between them.

Exploiting the co-occurrence of terms in information retrieval systems has been researched for a while [12–14], but our approach is different because it provides a computationally feasible mechanism for augmenting the text with discriminative c-features. Moreover, we consider that the co-occurrences of terms may happen in any portion of the document. As a consequence, our approach helps any classifier that assumes term independence in performing their task more effectively, as will be demonstrated by our experiments and characterizations.

We foresee four main contributions associated with this work: (1) an effective strategy for augmenting text based on term co-occurrence, making easier the construction of more accurate classification models, even for classes that comprise a small number of documents; (2) mechanisms that make the strategy computationally feasible, avoiding the combinatorial explosion associated with strategies that exploit co-occurrence; (3) quantification of the gains from employing both the strategy and the mechanisms for four reference collections, some of them widely known as challenging in terms of ATC; and (4) detailed analysis of the impact of the strategy on various algorithms (SVM [15,16], Naïve-Bayes [17], and *k*NN [1]) applied to real datasets.

The paper is organized as follows. Section 2 provides an overview of related work. Section 3 presents our approach for automatic text classification using data mining techniques. Section 4 discusses our experimental results, where four distinct reference text collections were employed: Reuters-21578, OHSUMED, 20 Newsgroups 18828 and ACM11. Section 5 presents a characterization of the impact of our strategy over different classifier algorithms. Finally, Section 6 presents the conclusions and outlines future work.

2. Related work

Research on automatic extraction of text features for improving text classification dates back to the early 1990s. Thanks to the availability of more powerful hardware and the challenge of classifying new text collections, text representations more sophisticated than the

[1] These are real examples that were actually generated in our collections by our proposed techniques, described later.

traditional *bag-of-words* have been investigated towards more effective text classifiers. In essence, these representations aim to reduce the inherent ambiguity of single terms, named *s-features*[2] from this point, and consequently, to build more powerful and effective classification models.

In the next sections we discuss some previous work about text-feature extraction for enhancing text classification. Sections 2.1–2.3 show distinct approaches of a common idea, which is the identification of semantic or syntactic relationships among terms and the use of these relationships in addition to unigrams for text classification. We present these approaches according to an evolutionary perspective. We begin by describing in Section 2.1 the earliest trials for acquiring features based on *syntactic phrases*. In Section 2.2 we discuss about works that use *word n-grams* (i.e., sequences of n adjacent terms) as additional features. Finally, in Section 2.3, we comment about previous work that uses non-adjacent terms that co-occur in any part of documents as additional features. The last approach is the basis of this work.

2.1. Syntactic phrases

The first efforts for acquiring features date back to the beginning of the 1990s when some researchers investigated the use of *syntactic phrases* as features for text classification [18–21]. Syntactic phrases have been defined as groups of words in the grammatical sense, e.g. noun phrases, verb phrases, gerund phrases, etc., and are obtained by some parsing process, for example, natural language processing techniques. Most of works that investigated the usage of syntactic phrases as features concluded that classifiers using them performed worse than those using the traditional bag-of-words approach. Lewis [18] explains the reasons for such failures, based on four desirable properties for text classification:

(1) Lack of redundancy among indexing terms.
(2) Relatively flat distribution of values for an indexing term.
(3) Lack of ambiguity for linguistically derived indexing terms.
(4) Low noise in indexing term values.

Lewis conducted experiments that showed that syntactic phrases tend to satisfy property 4, but not the others because (i) the frequency of syntactic phrases varied among classes and (ii) the ambiguity of the resulting models is significant as a consequence of using synonyms as a semantic equivalence criterion among syntactic phrases that are lexically different. In this case, considering that each s-feature within a syntactic phrase containing T terms has S synonyms on average, there are up to S^T syntactic phrases with the same meaning, which is a huge universe. Lewis concluded that the usually low frequency of semantically distinct phrases and the high dimensionality of the space of synonyms reduce the potential gains that syntactic phrases supposedly have as text features.

Later work [22] also did not achieve gains in classification by using syntactic phrases as features. Despite all these discouraging results, authors in [23] presented a study similar to that performed by Lewis, but they changed some implementation details in order to select the best phrases. Their strategy is to employ *rule-based algorithms*, based on the premise that such rules would consider the most discriminative syntactic phrases and induce better classifiers. Nevertheless, they evaluated their strategy using the collections Reuters-21578 and Digitrab, and the results showed that even the use of the best syntactic phrases did not outperform the bag-of-words strategy, but, instead, decreased the classification quality. However, the same classifiers, when combined through an ensemble, provided gains of 6% for Digitrab and 2% for Reuters-21578. The authors conclude that, considering the poor results, simple syntactic phrase-based representations are not worth further research.

Despite these phrase-based setbacks, there are several other works based on the use of sequences of s-features, as we discuss in the next section. These works focus on extracting sequences of s-features of varying sizes, but did not consider semantic analysis, identifying phrases that are morphologically distinct but are similar considering the mapping provided by synonyms and hyperonims. The works discussed in the next two sections employ this strategy.

2.2. Word n-grams

The authors of [24] investigated the use of word n-grams (i.e., sequences of n adjacent words/terms) of different sizes as features for text classification. They trained probabilistic classifiers using only frequent n-grams with at most five terms. Experiments showed that classifiers using n-grams composed by at most three s-features performed better than classifiers based on bag-of-words, but no improvement was obtained when using larger n-grams. However, only one text collection was used in the experiments and the classifiers based on bag-of-words that were used as baselines produced very low accuracies. In this scenario, it is usually easier to enhance the classifier's performance by using new features, being difficult to state whether the adopted approach would generalize to other text collections.

The work from [25] employs a similar approach, also using n-grams and their frequency within the documents. The strategy was evaluated using the collections Reuters-21578 and 20 Newsgroups, for which bag-of-words perform well. Results showed that, even considering the frequency of n-grams, no gains were achieved, and, in some cases, the strategy performed worse than bag-of-words.

Forman [26] reports good results for a strategy that employs just two-word sequences for classifying computer science articles. However, the experiments focused on classes that comprise just 2.8% of the documents in the

[2] Single terms are usually referred to as *unigrams*. However, we use the expression *s-feature* throughout the text to refer to a single-term feature in order to make a clear contrast with c-features (compound-features) as additional features to represent the documents.

collection, and even the author pointed that his study is not broad enough to draw general conclusions.

In [27], a stemming algorithm was applied on the s-features before determining the *n*-grams, aiming to consider some morphologically distinct *n*-grams as semantically similar. Also, differently from previous work, a given *n*-gram does not replace its respective s-features, but is used in conjunction with them. The rationale of this strategy is that, although *n*-grams could help disambiguating their respective s-features, many s-features still present discriminative power when considered in isolation. Among the 48 experiments using the Reuters-21578 collection, 28 of them showed that the combination of s-features and *n*-grams performed worse than when only s-features were used. Also, in the remaining 20 experiments, the combined scheme presented only marginal gains, showing that the adopted approach is not consistently effective.

In [28], authors also used *n*-grams and s-features in conjunction. However, they adopted a more rigorous criteria to select the *n*-grams, aiming at using only highly discriminative *n*-grams. As a result, only 2% of the *n*-grams (where $n=2$) in the collection were used in their experiments. The approach performed better than just bag-of-words, presenting statistically significant gains, although the baselines were not the best existing results in the literature. As a consequence, their results were worse than those obtained when using only bag-of-words together with more sophisticated classification algorithms. Anyway, the results achieved suggested that the combined use of s-features and high quality *n*-grams could produce better results if state-of-the-art classifiers were used. Analogously, in [29], a methodology using *n*-grams was used to classify e-mails, but without consistent gains, despite low baselines.

Bekkerman and Allan [30] conducted a similar study aiming at clarifying certain aspects of previous work based on *n*-grams, but using SVM as classifier. They also used the 20 Newsgroup collection, which is accurately classified using only bag-of-words features. According to the authors' conclusion, "the improvement is clearly statistically insignificant. However, this result is the highest (to our knowledge) text categorization result ever achieved on the 20NG dataset", that is, they could not be sure about the effectiveness of *n*-grams.

In summary, these various results show us that, although some of them report gains when using *n*-grams as features, these gains are only marginal or subject to specific circumstances. In particular, there has not been reported in the literature significant gains for well-known collections such as Reuters-21578 and 20 Newsgroups [30]. In that same paper the authors explain this difficulty by the fact that the baselines are already very high, being hard to improve. In this work we present a methodology capable of extracting new features, being able to achieve significant gains in both collections. We also tested our approach in two other collections of documents using three distinct classification algorithms, obtaining significant gains in all scenarios.

Notice that we relax the premises of the paradigms that make use of *n*-grams towards more effective features. For instance, we do not consider the order of occurrence of terms neither the distance among terms when we specify relationships among them, since two terms may be strongly related to each other despite the distance between them or the order they appear in the text [31]. Such sets of terms are usually referred to as *termsets* or *itemsets* [12] and an s-feature is a termset of size one. From now on, in order to distinguish better between single terms and termsets with more than one term, we will use *s-feature* to refer to single terms and use the expression *c-feature* (from *compound-feature*) when we refer to 2-term termsets. We only consider c-features containing two terms in this work because termsets of size two were shown to be good enough for disambiguating individual terms [32,33]. In the next section we discuss related work that uses c-features to enhance text classification.

2.3. Non-adjacent co-occurrences (c-features)

In [34], relationships among s-features are considered by comparing distinct phrases of the same document. The approach is based on the intuition that writers tend to emphasize relevant ideas by repeating some s-features in different phrases of the same text. However, in many collections, such as collections of abstracts of scientific papers, documents are small and the limited number of phrases per document is usually not sufficient to detect strong relations among s-features. For this reason, in our approach, we use the whole text of a document to find relationships among s-features in order to combine them into c-features.

Authors in [14] use frequent c-features to enhance the collection representation and also filtered out some c-features that had high frequency in different classes. One major difference between their work and ours is that they target multi-label classification while we focus on the single-label problem. However, their c-features-based classifiers did not provide consistent gains for any of the collections used (Reuters-21578 and OHSUMED) when compared to traditional classifiers that made use of only s-features.[3] Finally, it is hard to perform direct comparisons against their results, since they propose a pruning technique that reduces the impact of the exponential size of the termset search space, but did not provide the details necessary for implementation.

Rak et al. [13] presented a similar work but they associated frequency-related weights with c-features. Their argument was that the frequency of a feature is more important than its presence in some circumstances. It is interesting to notice that this work is complementary to ours, since, as we show in the results, we assess three independent factors, which quantify relevance, popularity, and vocabulary size, and define the set of c-features used, while they discuss weighting of c-features. We leave as future work to investigate weighting as a fourth factor,

[3] As we show later, we are able to outperform all baselines that consider just s-features.

although it is not clear to us whether it is independent from the factors that we evaluate here.

In [35], the authors show experimentally, using non-textual collections, that itemsets with medium-frequency are probably more discriminative and consequently more effective than low-frequency or high-frequency itemsets for text classification. In this work we show that low-frequency itemsets are also important in some circumstances in the text classification context. Also, all previous works used frequent c-features (or itemsets) of any size whereas in our work we use only c-features of size two. This is an important issue because the number of c-features of any size is exponential and, consequently, previous strategies just cannot be used with huge collections. On the other hand, the space of c-features with only two terms suffices to derive highly discriminative c-features as shown experimentally in this work. As a consequence, our approach represents a good balance between the computational feasibility and the classification quality, being effective in distinct collections with distinct characteristics and distinct levels of difficulty with respect to the classification task.

3. Feature extraction

In this section we describe our proposed methodology for feature extraction and also discuss some key implementation details to improve its performance.

3.1. Methodology

In this subsection we describe the overall methodology for extracting features for single-labeled collections. The intuition is to augment the document collection with c-features that capture co-occurrence information, improving the performance of classifiers that do not take into consideration such information, as illustrated in Fig. 1. Our methodology performs tasks during both the training and testing stages. We will start describing the tasks performed while training a classifier, which are divided into three steps.

The first step is enumeration, when we determine how single-features (individual terms) appearing in training documents are combined to generate c-features. It is possible to perform the enumeration using various strategies. We start in a bottom up fashion, creating pairs of s-features, and then combining those pairs with a different s-feature, creating triples, and so on. In this task, we employ two criteria to determine which combinations should be generated: the set of s-features and support. The first criterion determines which s-features are the best to be combined in order to generate relevant c-features. The second criterion is a significance measure for c-features, which defines the minimum number of documents where a c-feature should occur in order to be taken into account.

The second step is ranking, that is, determining the c-features that have better discriminative power, in the sense that they are indicators of a c-feature being strongly associated with a class. The ranking process is based on the *dominance* criterion. The *dominance* of a feature f in a class c is the conditional probability of c given the occurrence of f, estimated in the training set. Formally, let $F=\{f_1, f_2, f_3,\ldots f_M\}$ be the set of c-features associated with a collection; $C=\{c_1, c_2,\ldots,c_K\}$ be the set of categories that occur in a collection; $df(f_i, c_j)$ be the number of training documents associated with class c_j that contain f_i. We define *dominance* as follows:

$$dominance(f_i,c_j) = \frac{df(f_i,c_j)}{\sum_{j=1}^{K} df(f_i,c_j)}$$

The smaller the number of distinct classes where a c-feature occurs, the higher the *dominance. Dominance* is particularly interesting since it: (i) can be used to filter c-features distributed irregularly among several classes; and (ii) directly quantifies the pertinency of which classes should or should not have their documents augmented with a given c-feature.

The third step is the document augmentation, which aims to add c-features that help the classifier in performing its task. We first perform the augmentation in the training set. A key question here is to determine whether to include a c-feature to a document. We use dominance to determine whether to include a c-feature in the training set as we describe next. Suppose that a c-feature f is composed of s-features $f_1,f_2,\ldots,f_m$, $1 \le m \le |T|$, where T is the set of s-features. Let $C=\{c_1, c_2,\ldots,c_K\}$ be the set of classes that occur in the collection and let $dominance(f,c_j)$ be the dominance values of c-feature f in each class c_j of C. We insert the c-feature f in documents of the training set only if, given a class c_j, $dominance(f,c_j)$ is equal to or greater than a given threshold τ. Also, we include the c-feature f only in training documents that belong to the class c_j and that contain all of the s-features $f_1,f_2,\ldots,f_m$ that compose the c-feature f.

A c-feature f is also used to augment test documents if there is a class c_j whose $dominance(f,c_j)$ computed in the training set is equal to or greater than the threshold τ. However, since we do not know the category to which the test document belongs, we include the c-feature f in a test document whenever all s-features $f_1,f_2,\ldots,f_m$ that compose c-feature f are present in that document.

It is important to notice that in this work we are not concerned with how to define the best weighting methods for c-features given the very different and specific nature of this type of feature, when, for example, they occur in fewer classes with smaller dispersion. We believe that a study about weighting schemes specially designed for c-features is a work on itself and we suggest it as future work. Therefore, here we chose not to deal with this issue, using a common binary weighting scheme for both s-features and c-features, which would allow us to better focus on evaluating the fundamental reasons for the usefulness of c-features for classification.

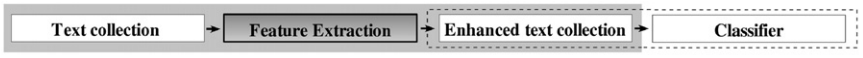

Fig. 1. Feature extraction overview.

Once we conclude the augmentation of training and test collections, we may employ any classification method, such as SVM, Naïve-Bayes, and *k*NN.

3.2. Implementation details

In this section we discuss some implementation details of our approach as well as practical issues that arise during the feature extraction.

One major issue is the combinatorial explosion associated with the c-feature enumeration, as a potential consequence of the high dimensionality of the data, since each s-feature is a dimension. As we discuss next, we employed a three-phase heuristic to reduce the computational cost in the first step of the methodology.

The first phase is to rank s-features for sake of enumeration according to their discriminative power in isolation. We used information gain (infogain) [36] as a criterion, but others may be applied. This strategy relies on the hypothesis that combining the most informative s-features extracts better c-features than if we combine low informative s-features. This hypothesis is confirmed in our experimental results. Consequently, this strategy makes it possible to define a smaller vocabulary subset to generate a considerable number of good c-features.

The second phase determines the portion of the sorted list of s-features that is used for building c-features. We then determine the value of a parameter N, which is the number of s-features considered in decreasing order of information gain. Empirically, we demonstrated that increasing N tends to generate more useful c-features and better classifiers. However, N has an upper bound, since using low discriminative s-features to extract c-features may not improve the classification, but may increase computational costs and add noisy c-features to the model.

The third phase is the enumeration of features. In this case, we enumerate just pairs of s-features, since the interaction between two terms is usually sufficient for term disambiguation [32,33], narrowing the possible meanings of one of the terms. Thus, given the whole training vocabulary of V s-features, and V is usually larger than N, we reduce the number of possible c-features that are added from $(2^V - V)$ to $N \times (N-1)/2$ (combinations of size 2), but still being able to improve the effectiveness of the classifier, as shown in our experiments.

A similar idea is used to verify the impact of the *dominance* factor in the feature extraction process. In order to select c-features characterized by having highly discriminative power, we have tested different values of threshold τ to decide about the inclusion of c-features in the augmentation step. This means that we only use a c-feature f in the augmentation step if there exists a class c such that $dominance(f,c) \geq \tau$. This approach meets our goal of emphasizing the intra-class relationships while minimizing the inter-class similarities.

4. Experimental evaluation

In this section we present results obtained with our feature extraction strategy. We first describe the collections and classification methods used and then the evaluation methodology.

4.1. Collections and methods

To experimentally validate our strategy, we run experiments using four different reference collections: 20 Newsgroups 18828, OHSUMED, Reuters-21578 and ACM11. In all collections, stopwords were removed as well as documents with multiple categories, except for the 20 Newsgroups 18828 which was already single-label.

The version of the 20 Newsgroups[4] collection (20 News) we used have 18,828 documents distributed almost uniformly among 20 newsgroup categories. The number of documents per class varies from 628 to 999. These documents are messages sent to newsgroups about themes such as science, religion, and politics.

The OHSUMED[5] collection contains medical documents collected in 1991 related to 23 cardiovascular disease categories. The version we used has 18,302 documents, distributed very irregularly among the categories varying from 56 to 2876 documents per category.

From the Reuters-21578 collection (Reuters), we used 8184 documents, representing news articles comprising merged content extracted from the tags "title", "author", "dateline" and "body".[6] As mentioned before, we removed documents with multiple labels. This resulted in a collection with eight categories, since two out of the 10 original classes had zero and one document, respectively, after the removal. The distribution in this collection is very skewed; the number of documents per class varies from 113 to 3930 documents.

The ACM11[7] collection is a sub-collection of the ACM Digital Library. All the text from the title and abstract, when available, was used to index documents. The original documents of ACM11 were not available, just their bag-of-words representation. The resulting collection comprising a set of 29,570 documents, labeled under the 11 categories of the first level of the ACM hierarchical classification system. Among all collections tested, ACM11 presents the highest class imbalance: the number of documents within classes varies from 93 to 7585 documents. Fig. 2 illustrates the distribution of document per classes for each collection presented.

We used the *k*NN and Naïve-Bayes classification methods, implemented in the *Rainbow* classification software [37], in our experiments. Besides these, we also used SVM-Perf [16], a package that implements a linear support vector machine (SVM) method, which may be trained in linear time. We used the one-against-all approach [38] in order to adapt the binary SVM classifier to multi-class classification, since our collections have more than two classes.

We employed these various classifiers in our experiments in order to demonstrate the comprehensiveness of our methodology, since each algorithm is based on

[4] http://people.csail.mit.edu/jrennie/20Newsgroups
[5] http://ai-nlp.info.uniroma2.it/moschitti/corpora.htm
[6] Exactly as performed by http://disi.unitn.it/moschitti/corpora.htm
[7] www.acm.org

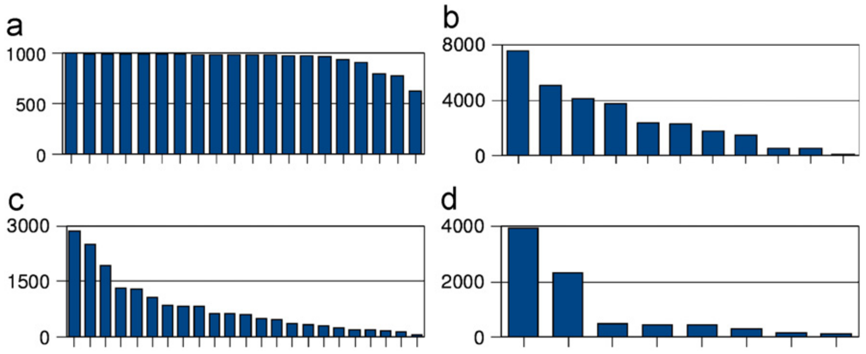

Fig. 2. Histogram of documents per class. (a) 20 News. (b) ACM11. (c) OHSUMED. (d) Reuters.

distinct approaches, varying from probabilistic to vector space. This also makes it possible to compare the relative effectiveness of our methodology among distinct classifiers applied over enhanced text collections.

4.2. Evaluation methodology

In this section we discuss the evaluation measures used in the experiments as well as their rationale in terms of the investigated parameter settings. In our experiments, we used a 70–30% random split for the training and test sets, respectively, in order to facilitate the comparison of the results of our feature extraction strategy over different collections with different algorithms. Also, the evaluation of the parameters of our method (i.e. the size of the vocabulary, dominance threshold, and minimum support), as discussed in the following subsections, was restricted only to the training set of each collection.

4.2.1. Evaluation measures

The effectiveness of our method was evaluated using standard information retrieval measures: precision, recall, and F_1 [39].

Let $C=\{c_1, c_2,\dots,c_K\}$ be the set of classes of a given collection. We first compute the following values for each class c_i:

- *True positives for* c_i (TP_i)—the number of test documents that the true class is c_i and that were classified in class c_i.
- *True negatives for* c_i (TN_i)—the number of test documents that the true class is not c_i and that were not classified in class c_i.
- *False positives for* c_i (FP_i)—the number of test documents that the true class is not c_i but were classified in class c_i.
- *False negatives for* c_i (FN_i)—the number of test documents that the true class is c_i but were not classified in class c_i.

Recall and precision can be computed per class or globally. The recall $r(c_i)$ for a given class c_i is defined as

$$r(c_i)=\frac{TP_i}{TP_i+FN_i}$$

in words, it is the fraction of test documents of class c_i that were correctly classified.

The precision $p(c_i)$ for a given class c_i is defined as

$$p(c_i)=\frac{TP_i}{TP_i+FP_i}$$

that is, the fraction of documents correctly classified from all documents attributed to the class c_i.

The global recall $r(C)$ is defined as

$$r(C)=\frac{\sum_{i=1}^{K} TP_i}{\sum_{i=1}^{k}(TP_i+FN_i)}$$

The global precision $p(C)$ is defined as

$$p(C)=\frac{\sum_{i=1}^{K} TP_i}{\sum_{i=1}^{k}(TP_i+TN_i)}$$

A usual approach to evaluate the classification effectiveness is F_1, a combination of precision and recall given by their harmonic mean, and is defined as

$$F_1=\frac{2pr}{p+r}$$

where r and p are the recall and the precision values, respectively.

We use two common scores to evaluate the results of our experiments. The first one is *micro-average* F_1, which is the F_1 measure computed over global recall and global precision:

$$\text{micro}-\text{average } F_1=\frac{2p(C)r(C)}{p(C)+r(C)}$$

The second score is the *macro-average* F_1, which is the average of the F_1 values for each class:

$$\text{macro}-\text{average } F_1=\frac{\sum_{i=1}^{K}\frac{2p(c_i)r(c_i)}{p(c_i)+r(c_i)}}{K}$$

Micro-averaged F_1 tend to be dominated by the classifier's performance on common categories and the macro-averaged F_1 are more influenced by the performance on rare categories [40].

If the classifier predicted none of the documents to a given class, we defined its F1 equal to zero.[8]

[8] However, in our experiments, this situation did not arise.

4.2.2. *Evaluating the size of the vocabulary for generating c-features*

The c-features are generated based only on higher infogain s-features of the training documents, where each document is defined as a transaction for the sake of feature extraction. This strategy is based on the assumption that combining the most discriminative s-features results in better c-features.

We verified this assumption by comparing the discriminative power between the c-features generated from the best half s-features ranked according to their infogain and those generated from the bottom portion of the same ranking. The results, in all of our reference collections, demonstrated that no c-feature generated by low infogain s-features were ranked among the higher infogain s-features, when we considered both types of features together. This demonstrates that s-features that are not good discriminants generate c-features which are not good either. On the other hand, more than 90% of the s-features at the top half in the infogain ranking left the top half ranking, being replaced by some of the generated c-features.

This finding demonstrates that not all s-features are useful for the sake of generating new relevant c-features. A natural question that arises is regarding the best size of this subset.

We then analyze the impact of the vocabulary size (N), sorted in decreasing order of infogain, for generating c-features. We inspect different values for N for each collection, as shown in Table 1. It is worth noting that when the vocabulary size (N) considered is equal to V, this is equivalent to using all the words to generate c-features.

It is also interesting to notice that, for the largest $N = 9.1 \times 10^4$, if we associate s-features in pairs (i.e., all possible two-by-two combinations), there are, theoretically, more than 4 billion possible c-features. However, given the sparsity of term co-occurrences in the documents, in practice we found that the number of c-features built is smaller than the theoretical bound by a factor of 100 or more. Further, this value shall still decrease by our process of discarding noisy c-features, which is presented next.

4.2.3. *Evaluating the discriminative power of the c-features*

The effectiveness of the c-features depends on whether they introduce more noise than useful information. Then, in order to investigate this effect, we run experiments using several levels of *dominance*. As discussed before, *dominance* is a measure that promotes c-features that will help to emphasize dissimilarities among classes. We tested six minimum *dominance* thresholds: 50%, 60%, 70%, 80%, 90% and 100%. Note that, in the last level, only c-features that occur in documents of just one class were generated. For the other levels, only c-features whose *dominance* is higher than the respective threshold were generated.

Another characteristic that may significantly influence the classification task is the frequency of the feature. According to Cheng [35], the use of infrequent features may prevent the model from generalizing well, since it is built based on statistically minor observations, which is referred to as over-fitting. Still according to that work, the discriminative power of a pattern is closely related to its support. On the other hand, many classification algorithms use *idf* as part of their strategy to weight feature's relevance, based on the intuition that the more documents a term occurs in, the less discriminative it is [1].

Nevertheless, in text collections, there might exist several small classes with predominantly low support features due to their relative small number of documents. Therefore, extracting c-features by selecting only those with high support tends to prominently help larger classes. Thus, this strategy also has biased potential. This leads us to test the hypothesis that low-support features could be important to increase the classification quality in small classes. We then defined a measure to quantify the minimum support (absolute frequency in the entire training corpus) which a c-feature must have in order to be selected to the classification model. We named this metric as *min_supp*, and we experiment with four levels: 2, 4, 6 and 8.

As explained before, we are not concerned about how to define the best weighting scheme for c-features. Therefore, in the experiments discussed next, we used binary weights for this type of feature as well as for s-features. Finally, for consistency and comparison purposes, with the bigrams approach, used as baseline, in our experiments (see next section) we also used binary weights.

4.2.4. *Experimental setup*

In summary, our experiments tested the three aforementioned factors: N, *dominance* and *min_supp*, using a "full factorial design experiment" [41] that considers all possible combinations of level values defined for each factor.

We tested this experimental configuration five times with different random training and test sets splits, for each of the four collections. Then, the classifiers were applied to each of the corresponding five sets enhanced with c-features, so that we could compare the average of these results to the results obtained with the original collections without c-features. In order to obtain a more precise comparison, the experiments without feature extraction were repeated 20 times. Then, a 2-tailed t-test with 99% confidence is applied to compare the results obtained using or not using feature extraction.

We also compared our results against a strategy based on n-grams, as proposed by [42]. In that work, n-grams are defined as sequences of n consecutive words (respecting sentences and related section boundaries) that appear in a document, which are used to augment

Table 1
Values of N for each collection used in our experiments.

Collection	Whole training vocabulary (V)	N		
20 News	$V_1 = 9.1 \times 10^4$	V_1	$\frac{1}{2}V_1$	$\frac{1}{4}V_1$
ACM11	$V_2 = 4.8 \times 10^4$	V_2	$\frac{1}{2}V_2$	$\frac{1}{4}V_2$
OHSUMED	$V_3 = 3.9 \times 10^4$	V_3	$\frac{1}{2}V_3$	$\frac{1}{4}V_3$
Reuters	$V_4 = 2.1 \times 10^4$	V_4	$\frac{1}{2}V_4$	$\frac{1}{4}V_4$

Table 2
Impact of feature extraction in macro-average F_1.

Macro-F_1	kNN macF$_1$ (%)			Naïve-Bayes macF$_1$ (%)			SVM macF$_1$ (%)		
Collection	bl.	bg.	f.e.	bl.	bg.	f.e.	bl.	bg.	f.e.
Reuters	83.7	85.7 +2.4 ▲	**87.9** +5.0 ▲	80.4	88.3 +9.8 ▲	**88.9** +10.5 ▲	91.5	91.4 −0.1 •	**92.7** +1.3 ▲
20 news	79.0	**89.1** +12.8 ▲	87.2 +10.3 ▲	88.0	90.6 +2.9 ▲	90.0 +2.3 ▲	90.1	90.3 +0.2 •	**91.9** +2.0 ▲
OHSUMED	61.7	59.0 −4.6 ▼	**64.7** +4.8 ▲	32.1	48.0 +49.5 ▲	**49.8** +55 ▲	61.2	62.8 +2.6 ▲	**67.3** +10.0 ▲
ACM11	50.8	–	55.2 +8.6 ▲	53.1	–	55.0 +3.4 ▲	56.9	–	58.8 +3.4 ▲

the original bag-of-words representation of the documents.[9] We implemented a suffix tree structure to store and manage n-grams, as proposed by [42], using bigrams ($n=2$) since they led to best results, also in that work. The goal here was to directly compare our approach against a similar one that did produce gains in specific scenarios.

4.3. Experimental results

We present here the results obtained in our experiments. Tables 2 and 3 present the classification results using macro-average F_1 and micro-average F_1 measures, respectively. In both tables, the column with heading "bl" (base line) presents values for classifications using only s-features, while the columns with headings "bg" and "fe" present values for classifications with the use of bigrams and our proposed feature extraction method, respectively. We represent in bold face the best results among the three methods. Each value for "bg" and "fe" is followed by the respective gain (in %), describing the relative variation in effectiveness due to classification with bigrams and feature extraction. Gains and losses are shown along with an indicator of whether the variations of using each methodology ("bg" and "fe" columns) with regards to the baseline ("bl" column) represent a statistically significant gain (upward triangle), loss (downward triangle) or equivalence (circle), given a 99% confidence in a 2-tailed t-test. In the tables we only do not have results for the bigram method applied to the ACM11 collection, since we did had just the bag-of-words representation of the documents, which is not enough to reconstruct the bigrams, due to the absence of word sequences and dependencies between sentence/sections delimiters.

Notice that, in these tests, we used the experimental configuration in which the levels of each factor produced the best results (seen in Tables 2 and 3) for each combination of algorithm and collection. This configuration, as discussed in Sections 4.3.2–4.3.4, was: (i) *Min_Supp* threshold=2, (ii) *size of vocabulary*=V for all collections and algorithms. For the *Dominance* threshold, it was, for SVM: 100% for 20 Newsgroups and Reuters-21578, 90% for OHSUMED and 70% for ACM11. For kNN: 80% for 20 Newsgroups, 90% for Reuters-21578, 100% for OHSUMED and 60% for ACM11. For Naïve-Bayes: 90% for 20 Newsgroups, 80% for Reuters-21578, 60% for OHSUMED and 70% for ACM11.

The first important aspect to notice from Tables 2 and 3 is that our feature extraction strategy resulted in classifier improvement in almost all of the tested scenarios, when compared to those that did not employ feature extraction. For macro-average F_1, there are statistically significant improvements in all 12 cases. Compared to the use of bigrams, our feature extraction strategy performed better in almost all cases, the exception being the kNN and Naïve-Bayes classifiers when applied to the 20 News collection, where bigrams were slightly better, and a statistical tie with SVM in the Reuters collection.

Considering micro-average F_1, there are statistically significant improvements when using feature extraction in 11 out of 12 results when compared to baseline, the exception being a statistical tie with SVM when applied to the Reuters collection. Considering bigrams, our feature extraction strategy also performed better in almost all cases, with one statistical loss with kNN in the 20 News collection and two statistical ties with Naïve-Bayes with 20 News and Reuters collections.

In order to further understand these results, we refer to [30], which analyzes the 20 News collection and states that it is a "simple" collection, where most of the terms are important for classification. In this case, bigrams will clearly be an important feature. This helps to explain the good performance of bigrams in that collection. However, in more complex collections such as OHSUMED and Reuters, this premise of equivalent performance of terms is not true, since only a few terms really matter for classification in both collections. In this case, the probability of consecutive occurrence of two terms which are both important for classification is smaller than their co-occurrence probability (being consecutive or not). Consequently, there is a higher probability of generating effective c-features than effective bigrams. This justifies the superior performance of our feature extraction in the other collections.

[9] It is worth noticing that in all experiments we conducted, either using bigrams or c-features, we preserved all s-features, which means that any extracted feature was used *in addition* to all s-features, not in substitution to them.

Table 3
Impact of feature extraction in micro-average F_1.

Micro-F_1	*k*NN micF$_1$ (%)			Naïve-Bayes micF$_1$ (%)			SVM micF$_1$ (%)		
Collection	bl.	bg.	f.e.	bl.	bg.	f.e.	bl.	bg.	f.e.
Reuters	91.7	92.5 +0.9 ▲	**94.4** +2.9 ▲	92.9	93.3 +0.4 •	**95.1** +2.3 ▲	96.0	96.0 0.0 •	96.4 +0.3 •
20 news	77.3	**89.4** +15.6 ▲	87.5 +13.1 ▲	88.7	91.0 +2.6 ▲	90.4 +1.9 ▲	90.3	90.5 +0.2 •	**92.0** +1.9 ▲
OHSUMED	67.9	65.7 −3.3 ▼	**71.4** +5.1 ▲	54.4	61.9 +13.8 ▲	**65.4** +20 ▲	69.9	70.8 +1.3 ▲	**74.1** +6.0 ▲
ACM11	63.2	–	71.2 +12.6 ▲	69.9	–	72.5 +3.8 ▲	71.8	–	73.3 +2.0 ▲

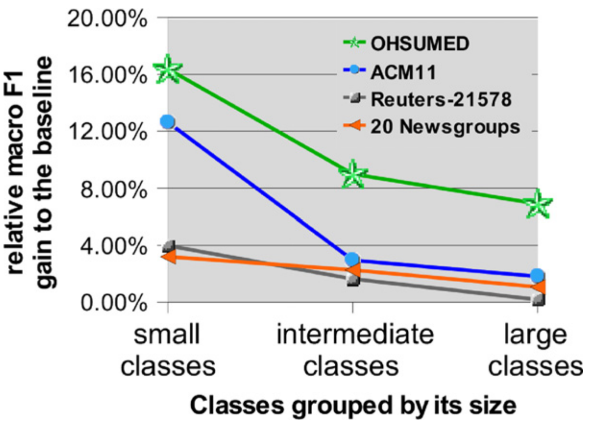

Fig. 3. SVM gains organized by class size.

An interesting point to stress according to Tables 2 and 3 is that larger gains were obtained in collections harder to classify, where the baselines using only s-features did not perform well. For example, for the ACM11 and OHSUMED collections, which are more difficult to obtain good classification results using only s-features, the SVM gains after using the proposed feature extraction strategy were more significant than for the other two collections.

Besides, even in the cases where the baselines' results are high and improvements consequently tend to be more limited, we achieved statistically significant gains. Likewise, for the best classifier observed, SVM, we achieved statistically significant gains in seven out of the eight results, the best one being a gain of 10.7% in macro-average F_1 in the OHSUMED collection.

4.3.1. *Analyzing the best results*

An important aspect to be considered is that, as we can observe by analyzing both Tables 2 and 3, the proposed strategy tends to improve more significantly the macro-average F_1 than the micro. This means that our strategy typically improves the classification quality mostly for smaller classes than for larger ones. Further evidence of this fact is shown in Fig. 3. In that graph, for each collection, we grouped its classes into three quantiles

Table 4
Comparing gains in small and large classes according to collection' imbalance.

Collection	Smallest class (%)	Largest class (%)	$\Delta gain$ (%)
ACM11	0.3	25.7	10.8
OHSUMED	0.3	15.7	9.5
Reuters	1.4	48.0	3.8
20 News	3.5	5.5	2.1

(small, medium, and large classes), based on the number of documents within each class. Then, for each quantile, we averaged the F_1 of its respective classes and compared this value to its respective baseline. The results clearly confirm our hypothesis.

These results are somehow counter-intuitive, since it is more likely that feature extraction would create more features from larger classes than from smaller ones. However, what happened is that the new c-features that were incorporated by our methodology helped to reduce the effect of the imbalance in the classes' size. In fact, the more imbalanced the collection, the more the smaller classes benefit from feature extraction. In Table 4, columns "smallest class (%)" and "largest class (%)" were measured by dividing the number of documents in the respective class by the number in the whole collection. The relative gains, "$\Delta gain$", were computed by subtracting the larger classes' average gains from smaller classes'. Table 4 shows that there is a direct correspondence between collection imbalance and the gains. One important reason for these results is due to the use of the minimum support factor (*min_supp*), which we analyze next.

4.3.2. *The impact of support variation*

The minimum support factor (*min_supp*) is a criterion to constrain the minimum number of documents in which a c-feature should occur in order to be selected for helping the classification process. Therefore, higher *min_supp* values may filter low-support c-features. However, patterns based on minor observations may be considered more significant to smaller classes, which, compared to larger classes, may be unable to generate high-support

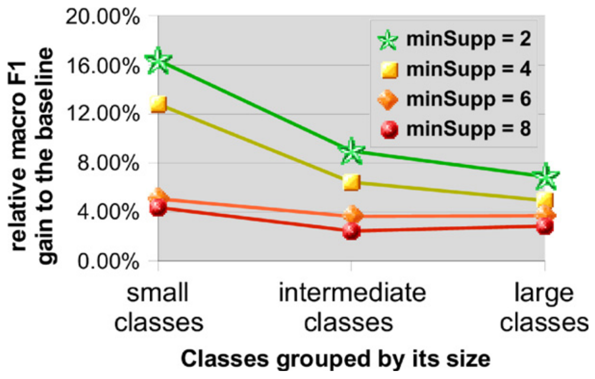

Fig. 4. SVM gains partitioned by class' size for each minimum support threshold. Collection: OHSUMED

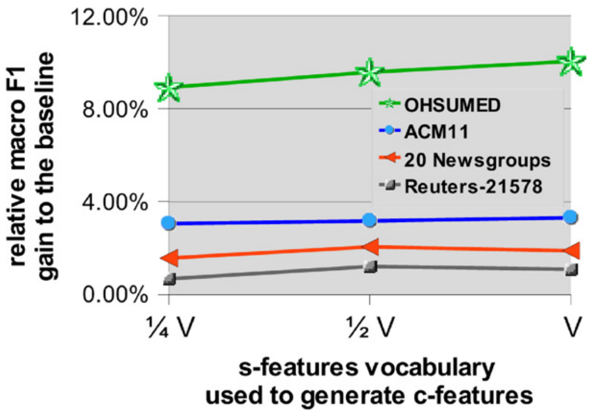

Fig. 5. SVM macro-F1 gains by s-features training vocabulary used to extract c-features.

c-features. Reinforcing this hypothesis, Fig. 4 reveals that low-support c-features are remarkably more critical to correctly classify documents in smaller classes. We use the OHSUMED collection to summarize these results, but similar results were obtained in the four reference collections. Further, this is in accordance to our hypothesis that low-support c-features may help to avoid over-fitting, given their property to reduce imbalance.

4.3.3. *Analyzing the vocabulary size—N*

In this subsection, we show that the best s-features with respect to infogain are good enough to generate high quality c-features. In order to support this argument, we conducted an experiment where we varied the number of s-features (N) considered, and calculated the gains obtained by using the resulting c-features compared to the baseline (classifier based on just s-features). We repeated this experiment for each collection and varied N from $V/4$ to V, where V is the number of s-features employed. The results of this experiment are summarized in Fig. 5, which shows that increasing N results in just marginal gains in the classifier effectiveness. We then may use a rather small N compared to $|V|$, and consequently reduce the number of generated c-features, without sacrificing classification accuracy.

4.3.4. *Analyzing dominance*

In this subsection, we analyze the effect of the *dominance*, the measure used for selecting good c-features. The idea behind this measure is to select only highly discriminative c-features in order to narrow the meanings of the original terms and induce better classification.

An important aspect about *dominance* is to discover a value that produces the best tradeoff between a smaller number of very discriminative c-features and a larger number of moderately discriminative ones. Unfortunately, there is not a single best global value for all collections. This occurs because, although a constant *dominance* value may select similar c-features, their impact on classification differs as a function of the collection characteristics. Further, different collections have very specific properties: (i) some lack highly discriminative words; (ii) others are more easily separable, containing several terms that may clearly characterize a class. From a practical standpoint, moderately discriminative c-features may help to improve classification in collections having the first property, but may be useless in collections having the second one. Different *dominance* values account for these differences in collection properties.

Table 5
Reference collection confusion.

Collection	Average infogain
Reuters	1.4×10^{-3}
20 News	0.9×10^{-3}
OHSUMED	0.8×10^{-3}
ACM11	0.3×10^{-3}

Our analyses show that for improving an already orderly collection, only higher discriminative c-features must be included. On the other hand, for a more chaotic collection, augmenting with a larger amount of moderately discriminative c-features may be more effective to improve the classification. To show this, we measured the average infogain of the s-features in the training collection to establish a relationship between it and the *dominance* value that better works for that collection. This information is shown in Table 5. This table shows that ACM11 is a relatively more “confusing” collection, while Reuters, 20 News and OHSUMED are relatively more “organized”.

As a consequence of this observation, it is more likely that only very discriminative c-features (higher *dominance* threshold) may be capable of improving Reuters, 20 News and OHSUMED classification. Nevertheless, for ACM11, the additional use of moderately discriminative c-features is better to enhance their classification. Fig. 6 validates this hypothesis. It shows that improvements associated with more organized collections are stronger when higher *dominance* thresholds are used, while less ordered collections improvements are better when using smaller *dominance* thresholds.

4.3.5. *Analyzing the computational cost*

One of the contributions of our study is a better characterization of the tradeoff between effectiveness and performance. This analysis is important because

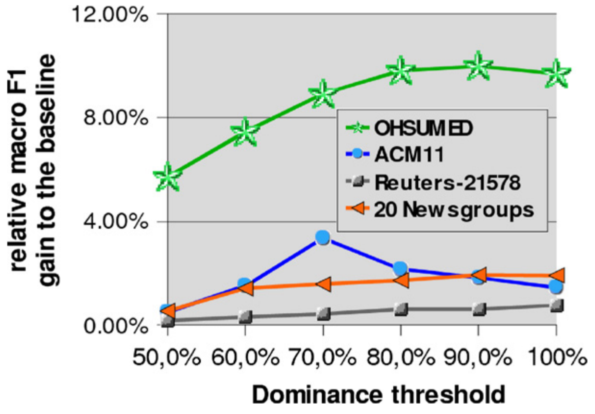

Fig. 6. SVM macro-F1 gains by varying *dominance* threshold values.

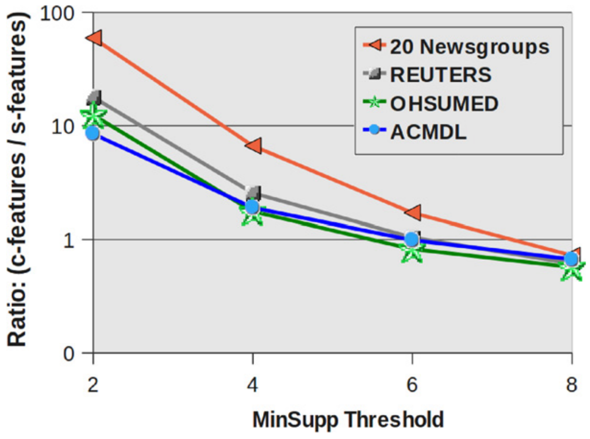

Fig. 8. Space cost by varying *Min_Supp* threshold.

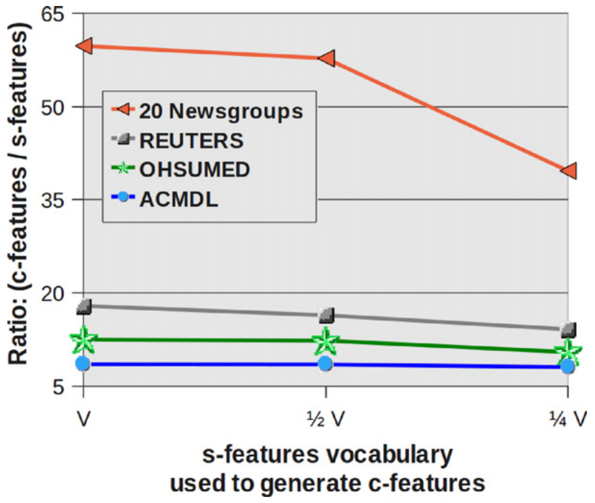

Fig. 7. Space cost by varying s-features training vocabulary used to extract c-features.

Table 6
Impact of feature extraction in micro- and macro-average F_1.

Macro-F_1	SVM			
Collection	bl. macF_1 (%)	bl. micF_1 (%)	f.e. macF_1 (%)	f.e. micF_1 (%)
Reuters	91.5	96.0	92.3 +0.8 ▲	96.1 +0.1 •
20 News	90.1	90.3	91.4 +1.5 ▲	91.6 +1.4 ▲
OHSUMED	61.2	69.9	64.0 +4.6 ▲	72.5 +3.7 ▲
ACM11	56.9	71.8	56.9 0.0 •	72.7 +1.3 ▲

methods similar to the one here proposed, that extract and add new features to the collections, may increase the computational cost of the supervised learning process. Thus, we present next a few analyses of the tradeoffs of our approach regarding space cost.

According to the characterization of SVM discussed in Sections 4.3.2 and 4.3.3, we observed that lower thresholds for *Min_Supp* and larger vocabulary sizes (V) produced better effectiveness. However, the choice of the most appropriate levels for these factors have also to consider the associated cost. Accordingly, we measured the tradeoff between space cost and different levels of these two factors. To measure space cost we used the ratio between the number of distinct c-features added to the collection and the number of distinct s-features already in the collection. Thus, for example, a ratio of 5 means that five c-features were added to the collection for each previously existent s-feature, meaning also a higher space cost. Figs. 7 and 8 show the relationship between space cost and the two aforementioned factors.

For instance, we can observe in Fig. 7 that the space cost in each collection can decrease, respectively, in average, from 3.4% and 19.1% when we use half or a quarter of the original vocabulary of s-features to generate c-features. However, if the collection is very large or the performance of the classifier is very affected by the space cost, further reductions are still possible when we vary the levels of the *Min_Supp* factor, as can been seen in Fig. 8. In these cases, reductions of more than an order of magnitude may be achieved. In fact, when we vary the *Min_Supp* factor, in average the space cost was reduced $6.9\times$, $19.0\times$ and $36.6\times$ when compared to using *Min_Supp*=2, when we used, respectively, *Min_Supp*=4, *Min_Supp*=6 and *Min_Supp*=8.

Table 6 shows the effectiveness results using *Min_Supp*=6. This is a specially interesting scenario in which the number of added c-features is approximately the same as the number of s-features for the four collections, as can been seen in Fig. 8. From Table 6, we can see that the proposed strategies can still produce statistically significant gains even without a large increase in space cost.

Finally, we present, for the sake of completeness, in Fig. 9, the variation of the space cost for different levels of *Dominance*. As we discussed in Section 4.3.4, the collections do not present a consensual pattern regarding the best levels of *Dominance* to be used. However, as can been seen in Fig. 6, low levels such as 50% or 60%, tend to not

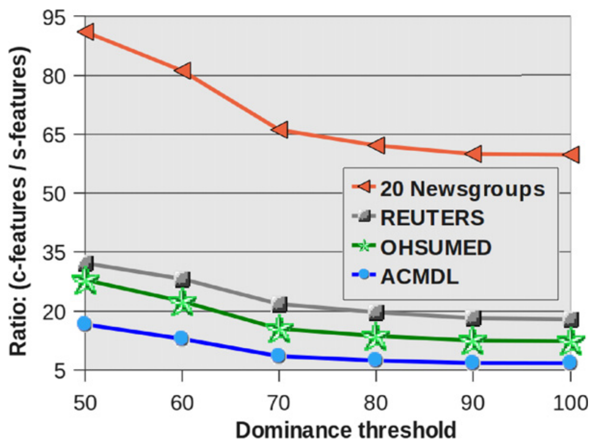

Fig. 9. Space cost by varying *dominance* threshold values.

produce good results. Besides, these are also the levels which impact more the space cost as can been seen in Fig. 9. However, after *Dominance*=70% the space cost ratio tends to stabilize in all collections.

4.3.6. *Summary of the analyses*

In summary, our analyses of the experimental results show that the cost of our approach is quadratic with the size of the vocabulary chosen for the construction of the c-features and that using half of the whole collection vocabulary is a feasible choice. In case of support, the collections tested suggest that the lower the values, the better. Finally, appropriate values for the dominance threshold may depend on the inherent confusion of the collection.

5. Discussion

The previous results show that the proposed feature extraction strategy is capable of improving classification effectiveness. In this section, we investigate further this issue by showing empirical evidence that some of our initial hypotheses were true, i.e., some c-features do really have higher discriminative power than some s-features.

5.1. *SVM*

Besides achieving the best results in our experiments, SVM provides us a general method for performing an investigation about features discriminative power. SVM with a linear kernel learns a linear function which gives weights for each s-feature or c-feature. High positive weights indicate that documents with these features are probably related to the positive class in a given binary classification, whereas negative features means they are to the negative class. In other words, these weights provide a method to "rank" the best features and allow us to compare the relative importance of s-features and c-features.

Table 7 displays some features' weights, from the highest absolute value to the lowest ones, for the class "pathological conditions, signs and symptoms", found in

Table 7
Features with largest and smallest weights as learned from the training data in the class "pathological conditions, signs and symptoms", from the OHSUMED's collection.

Type of feature	Weight	Feature
c-feature	0.167717	{postoperative, pain}
s-feature	0.160238	{deletion}
s-feature	0.150482	{graft}
s-feature	0.142706	{ventricular}
s-feature	0.136462	{bypass}
s-feature	0.135890	{arrhythmias}
…	…	…
c-feature	0.086127	{syndrome, premenstrual}
…	…	…
c-feature	0.077404	{palsy, facial}
…	…	…
s-feature	−0.068059	{tumor}
s-feature	−0.069713	{antibiotics}
s-feature	−0.081312	{cancer}
s-feature	−0.085188	{joint}
s-feature	−0.093884	{defect}
s-feature	−0.094051	{disease}

the OHSUMED collection. As can be seen from Table 7, the most important feature to discriminate the class in question is the c-feature "{postoperative, pain}".[10] Other c-features also appear with positive weights in the rank. More importantly, the weights for the s-features that compose the highest ranked feature are either very small (0.009611 for "pain"), or the feature was completely ignored by SVM (which was the case for the s-feature "postoperative"), an equivalent of a null weight. This suggests that feature extraction strategy is really creating helpful discriminative c-features for the classification task.

5.2. *kNN*

In order to evaluate the effects of the proposed feature extraction strategy over *k*NN, we inspected how the classification process of *k*NN was influenced by it.

In order to decide to which class c_i a test document d_j should be assigned, *k*NN looks at whether the k most similar training documents to d_j also are in c_i. As a result, *k*NN creates, for each d_j, a rank of classes. Moreover, since our collections are single-label, *k*NN assigns d_j to the best ranked class. In order to understand how the proposed feature extraction strategy changed this rank, we analyze how the rank position of the true class was changed. The same random training and test documents were used both for the baseline and with our strategy in order to avoid biased comparisons. We then defined a random variable X, which represents the number of positions in the rank gained or lost by the true class when using the proposed feature extraction strategy. For evaluating X, documents correctly classified by both, the baseline and our feature extraction strategy, were not considered.

[10] It is worth noting that in this category we found 59 documents containing the specific c-feature and that only in about half of them the respective s-features occurred in sequence, allowing it to be also discovered as a bigram.

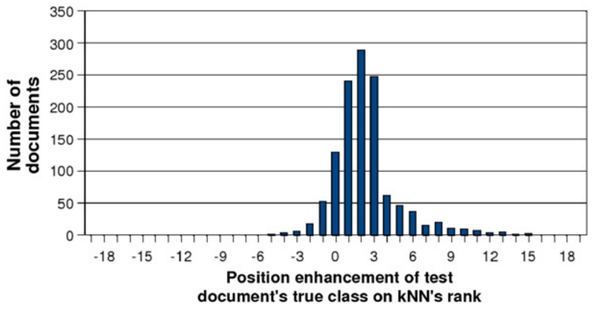

Fig. 10. Feature extraction effects over *k*NN: improvements on sampling process to correctly assign test documents to its true class. Collection: 20 Newsgroups 18828. Mean: 2.1; Skewness: 0.3.

Thus, more formally, we have

$$X(d_j) = Pos_{bl}(d_j) - Pos_{fc}(d_j)$$

where $Pos_{bl}(d_j)$ is the position of the true class for the test document d_j by using only s-features (baseline) and $Pos_{fc}(d_j)$ the analog as before, but also using c-features according to the proposed strategy.

Hence, we evaluated the random variable X for all reference collections tested. Fig. 10 shows its distribution for 20 Newsgroups 18828, which is similar for the other three considered collections. It can be seen that the mean value of the improved positions is greater than zero, meaning that our strategy enhances classification for most documents within the collection. Besides, we also quantified the skewness of the distribution in order to measure how asymmetric it is. We employed the third standardized moment [43] for this task. A positive skewness value means the distribution's right tail is longer than left one. Our measured skewness was positive (+3.4), which means that the distribution's right tail is asymmetrically longer. This fact, combined with the positive mean, demonstrates that the presented feature extraction strategy is capable of improving the ranking quality by bringing, in the class rank, the true class closer the top, even when it was far from it when using only s-features. Observing Fig. 10, there were many documents whose true class turns to rank expressively better, but few worse.

This effect was observed in all collections. Table 8 shows values found for mean and skewness for each collection tested.

5.3. Naïve-Bayes

Probabilistic classifiers are techniques that compute the probability of a document d_j to be assigned to the class c_i. The Naïve-Bayes algorithm uses Bayes's theorem to measure this probability, defined as

$$P(c_i|\overrightarrow{d_j}) = P(c_i)\frac{P(\overrightarrow{d_j}|c_i)}{P(\overrightarrow{d_j})}$$

The estimation of $P(\overrightarrow{d_j}|c_i)$ in the equation is troublesome, since the combinatorial number of possible vectors $\overrightarrow{d_j}$ is too high [1]. As a consequence, Naïve-Bayes mitigates this problem by assuming that every s-feature is independent. This allows to compute the probability of each term independently from the others, making classification computationally feasible.

Table 8
*k*NN: average rank enhancement.

Collection	Average enhancement of the position in *k*NN rank	Skewness
20 News	+1.3	+3.4
ACM11	+0.6	+2.7
OHSUMED	+0.6	+2.6
Reuters	+0.4	+2.2

Table 9
Naïve-Bayes: feature extraction influence on probability to document's true class.

Collection	Documents whose probability to true class improved (%)	Documents whose probability to true class decreased (%)
20 Newsgroups 18828	9.7	5.5
ACM11	34.7	15.2
OHSUMED	22.2	6.4
Reuters-21578	7.1	1.8

As a complement, our feature extraction strategy discovers discriminative relationships among terms, then breaking the independence assumption by introducing discriminative c-features.

In order to measure the impact of our strategy over Naïve-Bayes effectiveness, we measured how the added features influence $P(c_{true\_class}|\overrightarrow{d_j})$. As in *k*NN, the same random training and test documents were used to avoid biased comparisons. We then measured the ratio of documents whose $P(c_{true\_class}|\overrightarrow{d_j})$ gained or lost at least 1% towards its true class as a consequence of the proposed strategy. The results, presented in Table 9, show, for all collections tested, that there were more documents whose probabilities to its true class improved than the inverse. In sum, the proposed strategy affects Naïve-Bayes by boosting a given document's probability towards its true class.

6. Conclusions

We can distinguish two major factors that make document classification difficult. The first is the problem of defining the set of document features that better distinguish the class to which each document belongs. The second is related to the best method to be used in order to learn effective document classifiers, once the set of features has been defined.

In this paper we not only directly address the first issue, but also discuss how traditional classification methods can benefit from our strategy in order to generate better classifiers.

Specifically, we are concerned with extracting new features, named c-features, that are relevant to the classification task. The c-features are sets of terms that co-occur in any part of a document. We propose a feasible and effective method that allows us to obtain high discriminative c-features that are pairs of s-features (terms). The strategy has three steps. In the enumeration step we select the best s-features that will be used to build the c-features. In this work we used infogain as the primary criterion to rank the s-features and used just the top *N*. In the selection step we select the generated c-features that will be used to augment the documents of the training and test sets. As the selection criterion we use c-features with high dominance for a given category. Finally, in the augmentation step we insert in a given training document only those c-features that have high dominance in the class of the training document. The augmentation of a test document is done by inserting all high dominance c-features that appear in the document. In both cases, the dominance threshold is a function of the collection characteristics.

Experiments, conducted over the collections Reuters-21578, 20 Newsgroups 18828, OSHUMED and ACM11 using different classification methods, show that our feature extraction strategy consistently improves text classification. The most significative gains were obtained using the *k*NN classification algorithm, where we achieved up to 13% of micro-average F_1 for the Newsgroup collection. We also achieved gains up to 10.0% of macro-average F_1 using the SVM classifier in the OHSUMED collection, which is considered a state-of-the-art classifier. The experiments show that our proposed feature extraction strategy is effective regardless of the classification method being used. Moreover, it is a compromise solution that is able to obtain discriminative features while keeping the extraction process feasible. Furthermore, we noticed that the largest gains were verified in the collections harder to classify.

As future work, we suggest the investigation of other data mining techniques to exploit other relations among terms, for instance, the use of closed or maximal termsets to extract new features. Also, the use of c-features with sizes greater than two could be investigated. In this article we used the dominance criterion to select c-features to expand the representation of documents. Other distinct criteria could, as well, be investigated, for instance, the use of a feature selection technique to add only the best c-features. As stated earlier in the paper, we only consider a simple binary weight for both s-features and c-features. Another interesting work would be investigating issues related to weighting schemes for c-features, for example, defining a scaling factor for the c-features with respect of the corresponding s-features. Finally, it would be interesting to investigate how our approach can be generalized for collections in which documents are associated with more than one class.

Acknowledgments

This work is partially supported by CNPq, CAPES, FINEP, FAPEMIG and EconoInfo (www.econoinfo.com.br).

References

[1] F. Sebastiani, Machine learning in automated text categorization, ACM Computing Surveys 34 (1) (2002) 1–47.
[2] D. Sculley, G.M. Wachman, Relaxed online svms for spam filtering, in: SIGIR '07: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA2007, pp. 415–422.
[3] L. Zhang, J. Zhu, T. Yao, An evaluation of statistical spam filtering techniques, ACM Transactions on Asian Language Information Processing (TALIP) 3 (4) (2004) 243–269.
[4] W.S. Rongbo Du, R. Safavi-Naini, Web filtering using text classification, 2003, pp. 325–330.
[5] L.C. Mohamed Hammami, D.V. Tsishkou, Adult content web filtering and face detection using data-mining based kin-color model 1, 2004, pp. 403–406.
[6] K. Chandrinos, I. Androutsopoulos, G. Paliouras, C.D. Spyropoulos, Automatic web rating: filtering obscene content on the web, in: ECDL '00: Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries, Springer-Verlag, London, UK2000, pp. 403–406.
[7] T. Couto, M. Cristo, M.A. Gonçalves, P. Calado, N. Ziviani, E. Moura, B. Ribeiro-Neto, A comparative study of citations and links in document classification, in: JCDL '06: Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries, ACM, New York, NY, USA2006, pp. 75–84.
[8] G. Amati, D.D. Aloisi, V. Giannini, F. Ubaldini, A framework for filtering news and managing distributed data, Journal of Universal Computer Science 3 (8) (1997) 1007–1021.
[9] T.M. Mitchell, Does machine learning really work?, AI Magazine 18 (3) (1997) 11–20
[10] Y. Yang, J.O. Pedersen, A comparative study on feature selection in text categorization, in: ICML '97: Proceedings of the 14th International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA1997, pp. 412–420.
[11] G. Forman, An extensive empirical study of feature selection metrics for text classification, Journal of Machine Learning Research 3 (2003) 1289–1305.
[12] B. Possas, N. Ziviani, B. Ribeiro-Neto, W. Meira Jr., The set-based model for information retrieval, ACM Transactions on Information Systems 23 (4) (2005) 397–429.
[13] R. Rak, W. Stach, O.R. Zaiane, M.L. Antonie, Considering re-occurring features in associative classifiers, in: PAKDD'05: 9th Pacific–Asia Conference on Knowledge Discovery and Data Mining, Hanoi, Vietnam, 2005, pp. 240–248.
[14] O.R. Zaiane, M.L. Antonie, Classifying text documents by associating terms with text categories, in: CRPITS '02: Proceedings of the 13th Australasian Conference on Database Technologies, Australian Computer Society, Inc., Darlinghurst, Australia2002, pp. 215–222.
[15] T. Joachims, Making large-scale support vector machine learning practical, in: Advances in Kernel Methods: Support Vector Learning, MIT Press, Cambridge, MA, USA1999, pp. 169–184.
[16] T. Joachims, Training linear svms in linear time, in: KDD '06: Proceedings of the 12th ACM SIGKDD: International Conference on Knowledge Discovery and Data Mining, ACM Press, Philadelphia, PA, USA2006, pp. 217–226.
[17] P. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison-Wesley, Longman Publishing Co., Inc., Boston, MA, USA, 2006.
[18] D.D. Lewis, Representation and learning in information retrieval, Ph.D. Thesis, Amherst, MA, USA, 1992.
[19] D.D. Lewis, Feature selection and feature extraction for text categorization, in: HLT '91: Proceedings of the Workshop on Speech and Natural Language, Association for Computational Linguistics, Morristown, NJ, USA, 1992, pp. 212–217.
[20] D.D. Lewis, An evaluation of phrasal and clustered representations on a text categorization task, in: SIGIR '92: Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, NY, USA1992, pp. 37–50.
[21] C. Apté, F. Damerau, S.M. Weiss, Automated learning of decision rules for text categorization, ACM Transactions on Information Systems 12 (3) (1994) 233–251.
[22] S. Dumais, J. Platt, D. Heckerman, M. Sahami, Inductive learning algorithms and representations for text categorization, in: CIKM '98: Proceedings of the 7th International Conference on Information and Knowledge Management, ACM, New York, NY, USA1998, pp. 148–155.

[23] S. Scott, S. Matwin, Feature engineering for text classification, in: ICML '99: Proceedings of the 16th International Conference on Machine Learning, Morgan Kaufmann Publishers, Inc., San Francisco, CA, USA1999, pp. 379–388.
[24] D. Mladenić, M. Grobelnik, Word sequences as features in text-learning, in: Proceedings of ERK-98, the 7th Electrotechnical and Computer Science Conference, Ljubljana, SL, 1998, pp. 145–148.
[25] J. Fürnkranz, A study using n-gram features for text categorization, Technical Report, OEFAI-TR-9830, Austrian Institute for Artificial Intelligence, 1998.
[26] G. Forman, Feature selection for text classification, in: Computational Methods of Feature Selection, Chapman and Hall/CRC2007, pp. 257–276.
[27] M. F. Caropreso, S. Matwin, F. Sebastiani, A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization, 2001, pp. 78–102.
[28] C.-M. Tan, Y.-F. Wang, C.-D. Lee, The use of bigrams to enhance text categorization, Information Processing and Management 38 (4) (2002) 529–546.
[29] E. Crawford, I. Koprinska, J. Patrick, Phrases and feature selection in e-mail classification, in: P. Bruza, A. Moffat, A. Turpin (Eds.), ADCS, University of Melbourne, Department of Computer Science, 2004, pp. 59–62. URL 〈http://dblp.uni-trier.de/db/conf/adcs/adcs2004.html#CrawfordKP04〉.
[30] R. Bekkerman, J. Allan, Using bigrams in text categorization, Technical Report IR-408, Center of Intelligent Information Retrieval, UMass Amherst, 2004.
[31] N. Chomsky, Syntactic Structures, The Houge, Mouton, The Netherlands, 1957.
[32] A. Kaplan, An experimental study of ambiguity and context, Mechanical Translation 2 (2) (1955) 39–46.
[33] Y. Choueka, S. Lusignan, Disambiguation by short contexts, Computers and the Humanities 19 (3) (1985) 147–157.
[34] J. Feng, H. Liu, J. Zou, Sat-mod: moderate itemset fittest for text classification, in: WWW '05: Special Interest Tracks and Posters of the 14th International Conference on World Wide Web, New York, NY, USA, 2005, pp. 1054–1055.
[35] J.H.H. Cheng, X. Yan, C. Hsu, Discriminative frequent pattern analysis for effective classification, in: ICDE'07: Proceedings of 2007 International Conference on Data Engineering, Istanbul, Turkey, 2007, pp. 716–725.
[36] T. Mitchell, Machine Learning, McGraw-Hill, New York, NY, USA, 1997.
[37] A.K. McCallum, Bow: a toolkit for statistical language modeling, text retrieval, classification and clustering, software available at URL 〈http://www.cs.cmu.edu/∼mccallum/bow/〉, 1996.
[38] V. Vapnik, Statistical Learning Theory, John Wiley & Sons Inc., New York, NY, USA, 1998.
[39] D.D. Lewis, Evaluating and optimizing autonomous text classification systems, in: SIGIR '95: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, 1995, pp. 246–254.
[40] Y. Yang, X. Liu, 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, ACM, New York, NY, USA1999, pp. 42–49.
[41] R. Jain, The Art of Computer Systems Performance Analysis—Techniques for Experimental Design, Measurement, Simulation and Modeling, John Wiley & Sons, Inc., New York, NY, USA, 1991.
[42] R. Tesar, V. Strnad, K. Jezek, M. Poesio, Extending the single words-based document model: a comparison of bigrams and 2-itemsets, in: DocEng '06: Proceedings of the 2006 ACM Symposium on Document Engineering, ACM, New York, NY, USA2006, pp. 138–146.
[43] R. Groeneveld, An influence function approach to describing the skewness of a distribution, The American Statistician 45 (1991) 97–102.