Using Keyphrases as Features for Text Categorization

E. d'Avanzo, A. Lavelli, B. Magnini, and R. Zanoli

ITC-irst, Via Sommarive 18
38050 Povo, Italy
{davanzo,lavelli,magnini,zanoli}@itc.it

Abstract. *Keyphrases* provide a semantic metadata that summarize and characterize documents. Since keyphrases summarize documents very concisely, they can be used as a low-cost measure of similarity between documents, making it possible to group documents by measuring overlap between the keyphrases they are assigned to. In this paper we investigate the usefulness of keyphrases in *text categorization* (TC). We apply a keyphrase extraction algorithm to the Reuters Corpus Volume I (RCVI) and then we use the keyphrases extracted as features of the TC algorithm. We report the results of some experiments with different number of keyphrases per document comparing the performances with the *bag-of-words* (BOW) representation.

1 Introduction

The purpose of an automated information seeking system is to process information sources, and provide users with the information they need. The particular nature of an information seeking process is determined by the characteristics of information needs and information sources. For instance, *information retrieval* (IR) assumes a one-time user request and a dynamic collection. All seeking processes have in common a technique for representing needs and sources. Representation makes possible to automate the process of computing comparisons between need and sources. We are interested in this process, also known as *indexing* or *document indexing*.

Traditionally, IR has concentrated on assuming that a good representation for a document d may be obtained by taking into account whether and how frequently a word t appears in the document d and in the document collection, thus disregarding *morphological*, *lexical*, *syntactic*, and *semantic* variations.

In the *bag-of-words* (BOW) approach a text d_j is represented as a vector of weights $d_j = (w_{1j}..w_{rj})$, where r is the number of words that occur at least once in the document collection and $0 \leq w_{kj} \leq 1$ represents how much a word t_k contributes to the semantics of document d_j. Variants of the BOW approach are obtained by using *word stems* instead of words, or by disregarding frequency issues and simply using a binary assignment for w_{kj} based on either the presence or the absence of t_k in d_j (the *set of words* approach).

ITC-irst Technical Report, November 2003, 12 pp. Ref. No.: T03-11-01

1.1 The role of phrases

In the past many attempts have been done to use semantically richer features in IR tasks. In particular, a number of authors have investigated the use of phrases *instead of*, or *in addition to*, individual words, as features. According to [2] a phrase, in a linguistic sense, is *a textual unit usually larger than a word but smaller than a full sentence*. The definition includes either *noun phrases* or *verb phrases*. More specifically, the term *syntactic phrase* denotes any phrase that is such according to a grammar of the language under consideration. On the other hand, a *statistical phrase* is *any sequence of words that occurs contiguously in text*.

[2] have individuated a number of advantages of using statistical phrases with respect to syntactic ones:

- they may be recognized by means of more robust and less computationally demanding algorithms;
- the effect of irrelevant syntactic variants can be factored out;
- uninteresting phrases (e.g. *tall professor*) tend to be filtered out from interesting ones (e.g. *associate professor*).

On the other hand, *syntactic phrases* seem interesting because

1. the use of lexical atoms, such as "hot dog", to replace single words for indexing would increase both precision and recall;
2. the use of syntactic phrases, such as "junior college" to supplement single words would increase precision without hurting recall;
3. using phrases as index terms, a document that contains a phrase would be ranked higher than a document that just contains its constituent words in unrelated contexts.

In Section 4 we present some recent attempts to use statistical and syntactic phrases in *text categorization* (TC), i.e. the process of inductively learning to classify natural language texts with topical categories from a pre-specified set [7]. This paper reports some experiments performed using keyphrases for document indexing in the context of TC. The use of keyphrases as features is motivated by their ability of summarizing document concisely and in a *conceptual fashion*. Moreover, the use of keyphrases as features drastically reduces the number of features (*dimensionality reduction*), speeding up the TC task.

2 Text Categorization

Text Categorization (also known as *text classification*) is the activity of automatically building, by means of machine learning (ML) techniques, automatic *text classifiers*, i.e. programs capable of labelling natural language texts with thematic categories from a predefined set $C = \{c_1, ...c_m\}$.

A frequently used approach to building a text classifier for categories $C = \{c_1, ...c_m\}$ is that of building m independent classifiers, each capable of deciding whether a given document d_j should or should not be classified under

category c_i, for $i \in \{1...m\}$. This process requires the availability of a corpus $C_o = \{d_1^{'}....d_s^{'}\}$ of manually preclassified documents. A general inductive process (called the *learner*) automatically builds a classifier for category c_i by learning the characteristics of c_i from a *training set* $T_r = \{d_1^{'}....d_g^{'}\} \subset C_o$ of documents. Once a classifier has been built, its effectiveness may be tested by applying it to the *test set* $T_e = \{d_{g+1}^{'}....d_s^{'}\} = C_o - T_r$ and checking the degree of correspondence between the decisions of the automatic classifier and those encoded in the corpus.

2.1 Feature Selection

A major difficulty of TC is the high dimensionality of the feature space. The native feature space consists of the unique terms (words or phrases) that occur in documents, which can be tens or hundreds of thousands of terms for even a moderate-sized collection. Automatic feature selection methods include the removal of non-informative terms according to corpus statistics. More generally, feature selection may be modeled as follows (for further details see [2]). Let be r the length of the vectors that represent the documents, it is of key importance to work with vectors shorter than r. For this, *feature selection* techniques are used to select, from the original set of r features, a subset of r'$\ll r$ features that are most useful for compactly representing the meaning of the documents; the value $\rho = \frac{r-r'}{r}$ is called the *reduction factor*. These techniques consist in scoring each feature by means of a *feature evaluation function* (FEF) and then selecting the r' features with the highest score. Many functions have been used as FEFs in TC: *document frequency*, *information gain*, *mutual information*, χ^2 statistic, and *term strength* (for a comparison between the various methods see [11]). In this paper *keyphrases extraction* (KE) represents a kind of feature selection. Due to their summarization property, keyphrases allow a reduction of dimensionality comparable with standard feature selection methods.

3 Keyphrases

Keyphrases provide semantic metadata that summarize and characterize documents. They provide a brief summary of a document's contents and can be used in information retrieval systems as descriptions of the document returned by a query, as the basis for search indexes, as a way of browsing a collection, and as a document clustering technique. Keyphrases[1] are most familiar in the context of journal articles, but many other types of documents could benefit from the use of keyphrases, including Web pages, email messages, news reports, magazine articles, and business papers. Although the potential benefit of keyphrases is large, the vast majority of documents currently do not have keyphrases due to impracticality to assign them manually to documents. There are two general approaches to supply keyphrases for a document: *keyphrase assignment* and

[1] Or keywords, as they are sometimes called.

keyphrase extraction. Both approaches use supervised machine learning from examples. In both cases, the training examples are documents with manually supplied keyphrases.

In keyphrases assignment, there is a predefined list of keyphrases (i.e, a *controlled vocabulary* or *controlled index terms*). These keyphrases are treated as classes, and techniques from TC are used to learn models for assigning a class to a given document [3]. A document is converted to a vector of features and machine learning techniques are used to induce a mapping from the feature space to the list of keyphrases. The vectors are based on the presence or absence of various words or phrases in the input documents. Usually a document may belong to several different classes.

In keyphrases extraction, keyphrases are selected within the body of the input document, without a predefined list. When authors of journal articles assign keyphrases without a controlled vocabulary (*free text keywords* or *free index terms*), typically about 70% to 80% of their keyphrases appear somewhere in the body of their documents [8]. This suggests the possibility of using author-assigned free-text keyphrases to train a keyphrase extraction system. In this approach, a document is treated as a set of candidate phrases and the task is to classify each candidate phrase as either a keyphrase or non-keyphrase ([8], [10]). A feature vector is calculated for each candidate phrase and machine learning techniques are used to learn a model that classifies each candidate phrase as a keyphrase or non-keyphrase. The features include the frequency and location of the candidate phrase in the input document.

In this work we have used the keyphrases as features in a TC task. The keyphrases have been extracted using Kea [10], a package developed at New Zealand Digital Library. Kea uses the Naive Bayes machine learning algorithm for training and keyphrase extraction. The algorithm consists of two stages:

1. Training: it creates a model for identifying keyphrases, using training documents where the author's keyphrases are known.
2. Extraction: it chooses keyphrases from a new document, using the above model.

Both stages choose a set of *candidate phrases* from their input document, and then calculate the values of certain attributes (features) for each candidate. Kea chooses candidate phrases in three steps:

1. input cleaning;
2. phrase identification
3. case-folding and stemming

Two features are calculated for each candidate phrase and used in training and extraction:

- $TF \times IDF$, that measures the phrase's frequency in a document compared to its rarity in general use;

– *first occurrence*, which is the distance of the phrase's first appearance from the beginning of the document.

The training stage uses a set of training documents for which the author's keyphrases are known. For each training document, candidate phrases are identified and their feature values are calculated as described above. Each phrase is then marked as a keyphrase or a non-keyphrase, using the actual keyphrases for that document. This binary feature is the class feature used by the machine learning scheme.

To select keyphrases from a new document, Kea determines candidate phrases and feature values, and then it applies the model built during the training phase. The model determines the overall probability that each candidate is a keyphrase, and then a post-processing operation selects the set of best keyphrases.

4 Related Work

[1] have performed an evaluation of a linguistically-motivated indexing model. The approach taken by the authors is based on part-of-speech (POS) tagging and syntactic pattern matching. Different experiments have been performed with representation based on combinations of different POS categories. These representations combine elements belonging to the category of nouns with those of adjectives, verbs, and adverbs. The different representational choices are compared to the baseline (i.e. using all single words as index terms).

To evaluate the different indexing schemes the authors have measured the performance in a text categorization task. The experimental system is based on the vector space model, terms are weighted in a $TF \times IDF$ fashion, and classifiers are constructed automatically using Rocchio's relevance feedback method. The experiments have been performed on a dataset from Reuters-21578 text categorization collection using 90 out of 135 categories and all stemmed words as baseline. Table 1 summarizes the results discussed below. The experiments with unstemmed, stemmed and lemmatized words as index terms showed improvements in average precision less than 5%. The experiments based on indexing sets derived from combinations of part-of speech categories presented, as well, improvements over the baseline. Moreover, a reduction of the feature space of the 20.8% was obtained. The authors have concluded that the union of names and adjectives performs best, while the addition of verbs worsens the performance, and adverbs do not make any difference.

N-grams are another kind of index terms. In their work [9] support an attempt to improve categorization performance by automatically extracting and using phrases, especially two word phrases (hereafter *bigrams*). The bigrams extracted have been used in *addition to* (and not *in place of*) single words.

The experiments have been performed on two test corpora: a collection of web pages pointed to by the Yahoo! Science hierarchy and the Reuters-21578 corpus. In the following we will focus only on the results regarding the Reuters-21578 corpus, more similar to our experimental corpus.

Run	Average precision	Change	# Distinct terms	Reduction
Words	0.525	-2,2%	34,030	baseline
Stemmed words	0.537	baseline	27,205	20.0%
Lemmatized words	0.547	+1,9%	29,377	13.7%
Lemmatized nouns	0.559	+4,1%	23,039	32,3%
Lemmatized nouns and adj	0.563	**+4,8%**	26,952	**20.8%**
Lemmatized n, adj and v	0.540	+0,5%	24,977	26.5%

Table 1. Results of the experiments performed with part-of-speech tagging by [1]. In the last column the percentage of feature reduction wrt the baseline is reported.

Terms	Recall (%)	Precision (%)
Unigrams only	88.6	57.9
Bigrams only	95.9	6.9
Unigrams + Bigrams	90.8	57.6

Table 2. Recall and precision reported by [9].

To measure the performance the authors have used *recall* and *precision*. In particular, the break-even point (BEP) has been used, the point at which recall equals precision, and which is often used as a single summarizing measure for comparing results. Moreover, the F_1 measure has been used for evaluating the performance.

BEP increases in all categories under examination (12 out of 135), with the highest value at 21.4%. However, the performance as measured by F_1 was mixed. While the largest improvement remained at 27.1%, 5 out of 12 categories showed a drop. Table 2 shows the recall and precision rates before and after adding bigrams. The number of bigrams the algorithm finds is no more than 2% of the number of single words, so avoiding the problem of high dimensionality. Note that when bigrams alone were used, precision decreased drastically, while recall increased substantially. When both unigrams and bigrams were used, recall also improved, without significant decrease in precision. This means that bigrams are very good at identifying correct positives, but at the same time they introduced a large number of false positives, suggesting that bigrams should be used as the complements to the unigrams.

5 Experiments

For the experiments we have used the Reuters Corpus Volume I (RCVI), a set of documents recently made available by Reuters[2] for text categorization experimentation and consisting of 806,812 news stories produced by Reuters between

[2] `http://www.reuters.com/`

20-Aug-1996 and 19-Aug-1997. The stories cover the range of content typical of a large English language international newswire. They vary from few hundred to several thousand words in length. To aid retrieval from database products such as Reuters Business Briefing, category codes from three sets were assigned to stories: *topic codes*, *industry codes*, and *region codes*. The subset of RCVI chosen for our experiments consists of the news belonging to September 1996 (60,343 news: 40,229 news used as training set and 20,114 as test set). Only 101 categories (those related to *topic codes*) have been used.

As the learning device we have adopted ADABOOST.MH^{KR} [6], a more efficient variant of the ADABOOST.MH^{R} algorithm proposed in [5]. Both algorithms are an implementation of *boosting*, a method for supervised learning which has successfully been applied to many different domains and which has proven one of the best performers in text categorization applications so far. Boosting is based on the idea of relying on the collective judgment of a committee of classifiers that are trained sequentially; in training the k-th classifier special emphasis is placed on the correct categorization of the training examples which have proven harder for (i.e. have been misclassified more frequently by) the previously trained classifiers.

We will comply with standard text categorization practice in evaluating term categorization effectiveness by a combination of *precision* (π), the percentage of positive categorization decisions that turn out to be correct, and *recall* (ρ), the percentage of positive, correct categorization decisions that are actually taken. Since most classifiers can be tuned to emphasize one at the expense of the other, only combinations of the two are usually considered significant. Following common practice, a measure combining the two using their harmonic mean is adopted, i.e. $F_1 = \frac{2\pi\rho}{\pi+\rho}$. When effectiveness is computed for several categories, the results for individual categories must be averaged in some way; usually this is accomplished both by microaveraging and macroaveraging, defined in the usual ways (see [7]) and indicated by the "μ" and "M" superscripts, respectively. Microaveraging rewards classifiers that behave well on *frequent categories* (i.e. categories with many positive test examples), while classifiers that perform well also on infrequent categories are emphasized by macroaveraging. Whether one or the other should be adopted obviously depends on the application requirements.

All documents, in XML format, were preprocessed as follows:

- tag removing (Perl script);
- tokenization (Perl script);
- stopwords removing (Perl script);
- stemming (Perl implementation of Porter's stemmer).

We have evaluated both the impact of keyphrases in *place of* and *in addition to* BOW representation. To this end, we have performed different experiments using the following features as input to the TC algorithm:

- all terms (*set of words*, i.e. BOW without any weighting scheme) obtained from the preprocessing phase;
- only five keyphrases;

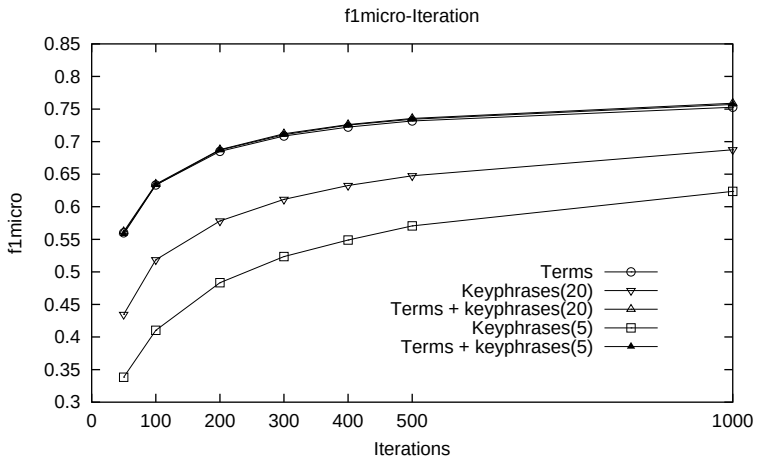

Fig. 1. Comparison of F_1^μ results with different features.

# of iterations	F_1^μ terms	F_1^μ terms + 5 keys	F_1^μ 5 keys
50	0.559979	0.558248	0.338300
100	0.633541	0.633467	0.410401
200	0.685070	0.687109	0.483355
300	0.708505	0.710543	0.523538
400	0.722225	0.725319	0.548906
500	0.731736	0.734407	0.570668
1000	0.752698	0.757056	0.623528

Table 3. Comparison of F_1^μ results with terms, 5 keyphrases only, terms plus 5 keyphrases.

- only twenty keyphrases;
- both terms and five keyphrases together;
- both terms and twenty keyphrases together.

Figure 1 shows the results of the F_1^μ with respect to the features mentioned above. Using only five keyphrases we obtained the worst performance due to a drastic drop in recall. The best performance was obtained considering as features both all terms (i.e. the set of words representation) and all terms with keyphrases together. Moreover, it should be noted that when adding either twenty or five keyphrases we obtained the same performance. Table 3 reports the results of the F_1^μ in detail. From the values reported in this table some negligible improvements may be noted when keyphrases are used.

Table 4 reports the results of the π^μ using all terms, only five keyphrases, and terms and five keyphrases together. The performance obtained using only

five keyphrases is comparable to the baseline. This represents an encouraging result because we achieve the same performance using only about the 33% of the original feature space (34,945 features using keyphrases with respect to 95,227 using all terms). Moreover, measures done considering single categories confirm the usefulness of keyphrases to improve precision. Table 5 shows the results of the precision with respect to a subset of the categories. On 24 categories out of 101 we have obtained the best performance[3] using only five keyphrases with an improvement of the average precision (macro-average) by about 30%.

A direct comparison with the experiments mentioned in Section 4 is not possible for two reasons: the use of different corpora, and the adoption of different evaluation measures. However, the trade-off between effectiveness and reduction of the feature space seems encouraging. We intend to investigate this further.

6 Conclusions and Future Work

In this paper we have reported preliminary experiments performed using keyphrases as features in TC task. Since keyphrases summarize documents very concisely, they can be used as a low-cost measure of similarity between documents, making it possible to group documents by measuring overlap between the keyphrases they are assigned to. The experiments performed so far seem to confirm the reasonableness of the hypothesis of exploiting keyphrases as a measure of similarity. In fact, performing the TC task in this fashion, i.e. using keyphrases as features, it is beneficial for a consistent reduction of dimensionality at a parity of π^{μ}. Experiments performed so far represent only our first attempt at considering these features in TC task. For future work we first plan to improve the algorithm of keyphrase extraction using different machine learning approaches. To this end, it would be desirable to adopt an unsupervised learning method to avoid the bottleneck of using annotated corpora. Second, we plan to use keyphrases in addition to other feature selection methods to avoid an increase in the dimension of the feature space. To this end, we will perform experiments using keyphrases and n-grams together. This mixed approach is motivated by the fact that the performances of keyphrases and n-grams seem to complement each other (keyphrases increase precision, while n-grams emphasize recall). Finally, we plan to test our results in web-oriented applications. In the Infomine Project[4] there is an interactive aid for keyword extraction, designed to assist a human classifier. In particular, it introduces a novel keyphrase extraction heuristic for web pages which requires no training. This is a stimulus to perform further experiments in order to find more effective features in this fashion.

[3] More precisely, we have obtained the best performance on 38 categories out of 101. However, in the categories not counted (i.e. 14) the results using five keyphrases are the same as those using twenty keyphrases.

[4] `http://www.infomine.ucr.edu/`

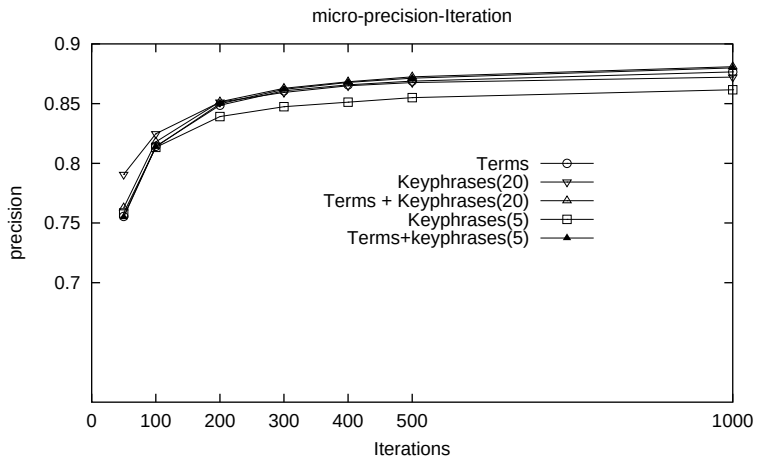

Fig. 2. Comparison of precision results with different features.

# of iterations	π^{μ} terms	π^{μ} terms + 5 keys	π^{μ} 5 keys
50	0.755570	0.754952	0.758405
100	0.814214	0.813813	0.813348
200	0.848674	0.849857	0.839163
300	0.860657	0.861889	0.847425
400	0.865678	0.867744	0.851208
500	0.868900	0.871358	0.854969
1000	0.876564	0.879923	0.861648

Table 4. Comparison of the precision results with terms, 5 keyphrases only, terms plus 5 keyphrases.

Acknowledgments

We thank Fabrizio Sebastiani for providing the AdaBoost.MH^{KR} classifier.

References

1. Arampatzis A., van der Weide Th. P., and Koster C.H.A., van Bommel P.: An Evaluation of Linguistically-motivated Indexing Schemes. Proceedings of the BCS-IRSG, 22nd Annual Colloquim on Information Retrieval Research. Cambridge. (2000).
2. Caropreso, M. F., Matwin, S., Sebastiani, F.: A Learner-Independent Evaluation of the Usefulness of Statistical Phrases for Automated Text Categorization. In Amita G. Chin (ed.), Text Databases and Document Management: Theory and Practice. (2001)

Category	π terms	π 5 keys	Improvement (%)
C13	0.554945	0.705882	27.2
C18	0.765120	0.817259	6.81
C42	0.767773	0.892308	16.2
E41	0.802372	0.951613	18.6
GPOL	0.772959	0.848404	9.7
C11	0.504587	1.00000	98.2
C15	0.867525	0.915254	5.5
C152	0.831407	0.902752	8.5
C33	0.800000	1.00000	25.0
GPRO	0.666667	1.00000	49.9
E51	0.643243	0.851852	32.4
M13	0.805195	0.836134	3.8
M132	0.755647	0.913725	20.9
E512	0.685897	0.805195	17.7
C12	0.800000	1.000000	25.0
C173	0.818182	1.000000	22.2
GDEF	0.636364	1.000000	57.1
E31	0.615385	1.000000	62.5
E513	0.863636	1.000000	15.8
C183	0.746835	1.000000	33.9
G151	0.327586	0.888889	171.3
G154	0.750000	0.937500	25.0
G155	0.300000	0.500000	66.7
G152	0.571429	1.000000	75.0

Table 5. Precision results for some of the categories.

3. Dumais, S., Platt, J., Heckerman, D., Sahami, M.: Inductive learning algorithms and representations for text categorization. Proceedings of the Seventh International Conference on Information and Knowledge Management. ACM Press (1998)
4. Lewis, D.D.: Evaluating and optimizing autonomous text classification systems. Proceedings of SIGIR-95, pp. 246-254. ACM Press. (1995)
5. Schapire, R.E., Singer, Y.: BoosTexter: A Boosting-based System for Text Categorization. Machine Learning, vol. 39, 2/3, 135-168. (2000)
6. Sebastiani, F., and Sperduti, A., Valdambrini, N.: An improved Boosting Algorithm and its Applications to Text Categorization. CIKM (2000)
7. Sebastiani, F.: Machine Learning in Automated Text Categorization. ACM Survey. (2002)
8. Turney, P.: Extraction of Keyphrases from Text: Evaluation of Four Algorithms. National Research Council, Institute for Information Technology, Technical Report ERB-1051. (1997)
9. Tan Chade-Meng, Wang Yuan-Fang, Lee Chan-Do: The use of bigrams to enhance text categorization. Information Processing and Management, Vol. 38. (2002)
10. Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C., Nevill-Manning, C.G.: KEA: Practical automatic keyphrase extraction. Proceedings of Digital Library 99 (DL '99). ACM Press. (1999).

11. Yang, Y., Pedersen, J.: A comparative study on feature selection in text categorization. In Proceedings of ICML-97, 14th International Conference on Machine Learning. Nashville. (1997)