Segmentation-based Feature Selection for Text Categorization

Zsolt Minier Zalán Bodó Lehel Csató
Department of Mathematics and Computer Science
Babeş-Bolyai University
RO-400084, Cluj-Napoca, Romania
Email: {minier, zbodo, lehel.csato}@cs.ubbcluj.ro

Abstract

Text categorization is an interesting problem in artificial intelligence that gets more and more attention from researchers and industry. One central problem of text categorization is the selection of a good feature set. We propose a novel method for term selection for each category based on segmenting the documents belonging to a category into cohesive sub-parts that define the subtopics of the document. Next we cluster these segments and use the terms found in the biggest segment cluster for each category. We compare the performance of our method with a very efficient ranking technique (χ^2) and find it very similar.

1. Introduction

These days the rapid growth of the internet makes available an amount of textual data that can not be processed manually; fast and *good* computer algorithms are researched in order to efficiently process the available data.

One of the tasks that arise naturally is the classification of the text contained in documents, meaning the assignment of one or more predefined labels to a document, based on the semantics of the text it contains.

The prevalent approach to classification of documents is based on the machine learning paradigm. This consists of providing a learning algorithm with a learning corpus and based on this corpus the algorithm extracts "some information" that will be used to categorize unseen documents.

The task of learning can be divided into two major subtasks. The first is that of the representation of the text in a way that can be handled by various machine learning algorithms; we can say that we extract *features*. The second task is the actual learning of the classes, *i.e.* to compute an abstract representation for the class based on the features of the documents that belong to it.

Both issues are of equal importance because a bad representation can degrade the performance of the learning algorithm while a good learning algorithm can still perform well even if some terms are redundant or a few important ones are missing.

Testing of the methods is usually done by assigning a set of categories to a testing document for which the correct set of categories is known in advance, and intersecting the two sets.

Considering the results in the survey of [18] it can be seen that the best text categorization systems are performing at an equal level although they apply different learning algorithms. Thus our aim is finding a better representation of the documents.

The structure of the paper is as follows: in the next section we outline the existing approaches to represent documents in information retrieval systems with some notes on performance evaluation. Section 2 presents our method going into details on each phase of the algorithm. The paper ends with a section containing the experimental results and a section enumerating future improvements regarding our method.

1.1. Document representation methods in text categorization

Most of the present day categorization methods rely on a word based vector space representation of documents. In the early work of [11] it has been argued that there is no need of representing text with more complicated features than words. The first experimental hypothesis of Lewis is:

> "A purely phrasal indexing language will have a lower optimal effectiveness than a purely word-based indexing language, and its optimum effectiveness will occur at a larger feature set size."

He proves this hypothesis based on his experiments on the Reuters corpus according to which more terms based on phrases were needed to reach the optimal performance which was even slightly lower than the performance obtained using word based terms. To explain his results Lewis

argues that a phrasal indexing language has inferior statistical properties compared to a purely word based indexing language. Thus more phrasal terms are needed to convey the same meaning because an important word can appear in the neighborhood of many different words.

However, it is clear that representing documents based only on the appearance of certain words in their content entirely disregards any kind of semantic structure of the documents transforming them into a bag-like structure. Still, as this method works quite well in practice it became the standard, so in most systems, a document is represented as a vector of word occurrences:

$$\vec{d_j} = (w_{j1}, ..., w_{ji}, ..., w_{j|T|})$$

where T is the set of terms that were chosen to represent the documents. The weight w_{ji} measures the "degree" to which the word i is important in the document d_j.

The number of terms in the document vector is an important issue because some learning algorithms can not handle vectors with high dimensionality, and also the problem of overfitting arises. The number of terms best representing the semantics of a document is not obvious because a small number of terms does not include enough information about all the possible phenomena that can be related to a category of documents, while too many terms loose generality and lead to the inclusion of non related terms that increase confusion.

Thus, experience shows that the best performance is usually reached at a fraction of the original number of terms. An exception is the case of SVM learning: it has been shown by [8] that SVMs handle well the high dimensionality and the sparseness of the problem.

To reduce the dimension of the term set, dimensionality reduction is done. This can be done in several ways. One way is clustering the terms, another method is using numerical methods like SVD. The easiest and most used method is ranking the terms according to measures that were found to discriminate well between important and unimportant terms.

In [22] there are enumerated the most common methods used for term ranking, like document frequency of the words, the χ^2 statistic metric between words and categories, or information gain, an entropy-based measure, or mutual information which is often used in the statistical theory of dependency grammars. They found χ^2 statistic to be the best method, and this is also in line with our experience, so we used it to determine our baseline performance.

Measuring the performance of a text categorization system is done by measuring its precision and recall. These measures are defined as

$$\text{precision} = \frac{\text{correctly found categories of documents}}{\text{all document categories found}}$$

$$\text{recall} = \frac{\text{correctly found categories of documents}}{\text{all existing categories of documents}}$$

However, reporting only precision and recall results will not characterize the systems' performance, because for example a classifier that says "yes" to every $\langle \text{document}, \text{category} \rangle$ pair will get a perfect recall score, but a very low precision. Thus, usually the composite F_1 measure is used that balances the two measures ([17, Ch.7]):

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

Another measure that is commonly used is the breakeven point. This is calculated by tuning the parameters of the classifier algorithm that are responsible for ranking the possible categories to be assigned. These parameters are tuned to the point where precision and recall are equal for each category. Of course this point might not exist, and in this case the average is taken between the precision and recall levels where these two are closest to each other.

Results can be microaveraged when the number of correctly found categories for a document is divided by the total number of categories found, or macroaveraged when performance is calculated for each category independently, and these results are averaged.

We report our results giving both microaveraged and macroaveraged precision, and recall levels, and also the F_1 measure, and breakeven point.

There were recent approaches to reach beyond the bag-of-word approach. In [13] a string kernel is considered that computes the substring based measure of two documents, thus effectively solving the problem of mistyped words and lemmatization. However, their method is unable to handle big corpora straightly, thus approximation methods are needed that degrade the precision. Another experiment of phrasal terms was done by [6], but no improvement in the results was observed. They used a Naïve Bayes classifier.

In [7] interesting results on their method of feature generation for the bag-of-words model are reported. They argue, that augmenting the document vectors with real world knowledge solves the problem of words that are important for the discrimination of testing documents, but are too rarely seen in training to be assigned the appropriate importance. Even with considering only generated terms using outside information during training they get similar performance to the best results using inner information from the corpus. Still, by using the augmented document vectors that are much larger than those built only from the same corpus we used for experimenting, they only gain 0.3% in the microaveraged breakeven point measure, however they gain 2.0% in the macroaveraged breakeven point measure proving that their technique did help the categorization of smaller categories whose terms are rare. One of their key insights is the segmentation of documents into smaller parts,

and analyzing the terms in these chunks because usually a document contains text on more than one topic. However, the chunking of the documents is done on strict levels and is done because of context considerations for their terms.

2. The proposed method

Our method still uses the bag-of-words model, however the term selection technique tries to take into account the semantics of the documents. We feel that the semantic structure of the document should be taken into account, as there is much unexploited potential there. One such feature of the semantic structure of a document is the sequence of topics it touches. Usually the text found in a document is the sequence of semantically loosely connected sentences each of which can be individually about a different topic making up together the meaning of the document. Thus many smaller segments of a document can resemble parts of documents contained in totally different categories.

An example document (number 0012007) from the *copper* category of the test set of the Reuters-21578 corpus:

CIPEC STUDYING COPPER MARKET BACKWARDATION

PARIS, June 29 - The Paris-based Intergovernmental Council of Copper Exporting Countries (CIPEC) is closely studying the current backwardation in world copper market prices but does not envisage taking corrective action at present, CIPEC sources here said.

The organisation's executive and marketing committees reviewed the current market situation during a series of meetings here late last week, but took no major decisions.

The sources noted that the backwardation - premium of nearby supply over forward delivery - dates back several weeks and is the longest on record.

"It's unusual," one official said, but added CIPEC did not have any immediate recipe to remedy the situation.

The meetings featured a gathering of the 10 directors of CIPEC's regional copper development and promotion centres, which are based in Europe, Japan, India and Brazil.

Their main aim was to prepare the ground for the annual ministerial meeting of CIPEC, which is scheduled for Zaire in late September.

The last three ministerial meetings have been held in Paris to keep down costs.

First let us note that the word *CIPEC* can not be found in the training part of the corpus, and even in the test set only in this document. In this document only the first and fifth sentence has some explicit information about copper (meaning that it includes the word copper) while the rest of the document could be about any kind of material with its market in trouble. However, many of the words present in the uninteresting part of document are important words in identifying other categories, such as *Brazil* for example that has a high density in documents about cocoa (16.36%), coffee (49.54%), orange (43.74%), and ship (14.21%). The occurrences in these four categories consist 36.67% of the total occurrence of the term *Brazil* in the training corpus and there is not one occurrence of it in the *copper* category. So, when classifying a test document, it is likely that it will be in one of these categories, so this term might be considered important for these categories and also unlikely that we find this term in a document belonging to the *copper* category. This way the term *Brazil* outweighs *copper* and results in an incorrect categorization.

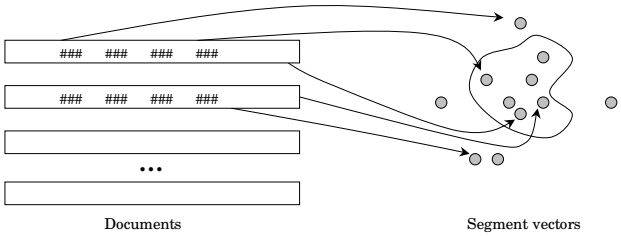

Figure 1. The segmentation and clustering process. The documents are segmented into cohesive parts, and then the segment vectors are constructed and clustered in each category.

Our idea is that the documents should be segmented into semantically similar cohesive parts and these segments have to be used when deciding on the representation of the documents when learning and testing. Thus we segment every document into semantically similar parts, and then these mini-documents – called *segments* or *cohesive parts* – are represented as word vectors.

To segment the documents we use a metric of semantic similarity of sentences. There are many such metrics, however we used one of the simplest ones: that from [4]. Once we have a semantic similarity metric between sentences we proceed by grouping sentences based on their order in the document. As long as the similarity of two consecutive sentences is above a threshold they belong to the same segment. When the threshold is not met a new segment is started beginning with the second of the evaluated sentences. In this way, semantically similar sentences can be grouped into different segments if there is a semantically dissimilar sentence between them but we let the final clustering of the groups handle this problem.

In the learning phase, after we segmented all the documents in a category we cluster the segments. We believe that the clustering should result in one large and many smaller clusters. We believe that the largest cluster will contain the most relevant words for the category because most of the segments in a category should be discussing the topic of that category. Those segments that are not sharing many words with the majority of the segments will form smaller clus-

ters and will be discarded. From the largest cluster we can extract the terms describing the respective category by simply taking the words appearing in the segments belonging to this cluster. After finding the term set for each category, the total term set is constructed by joining them together. Then we train a classifier on the document vectors constructed using these terms.

In the testing phase classifying an unseen document is done in the usual way, these documents are not segmented.

In the following we briefly describe the details of our method first the segmenting and clustering used to extract the terms, and then support vector machines as the used classification method.

2.1. Semantic similarity of sentences

Our aim is to segment a document into cohesive parts based on a sentence similarity measure developed in [4]. The procedure is quite simple, we calculate the similarity of successive sentences in a document and until this value falls below a predefined threshold the sentences form a coherent segment. Our idea is similar to the one described in [10].

The sentence similarity measure introduced in [4] is based on word semantic similarity metrics using WordNet. Comparing the semantics of two sentences taken from different contexts and disregarding those is conceptually impossible because sentences are as ambiguous as words. Just remember the example pleaded by [9, p.4] where the five different meanings of the sentence "*I made her duck*" were enumerated. But if the sentences under review are consecutive parts of a semantically interdependent continuous text then the probability that two neighboring sentences are far from each other in meaning and yet are *lexically* similar is very small (except NLP articles) and the afore-mentioned segmentation can be performed by observing only the separate sentences.

Human differentiation of concepts is based on semantic relations between these. WordNet is such a semantic network initially developed to provide conceptual searching and it has been proven to be a very beneficial resource in various domains such as natural language processing and information retrieval. WordNet contains only content-words, that is nouns, verbs, adjectives and adverbs, stored apart and connected by different types of semantic relations. The concept got the name *synset* because it is represented by a synonym set. Relations include synonymy, antonymy, hyponymy/hypernymy (IS-A), meronymy/holonymy etc. The word semantic similarities developed so far take into account the so engendered hierarchical structure of the relations and some properties of this network like density, relation type, depth and link strength. The semantic similarity metrics can be categorized into several classes by the means they calculate similarity: edge-counting, information content, feature-based and hybrid methods. A good enumeration and brief description of most of these measures can be found in [20]. In our experimentation we used the Perl package Wordnet::Similarity developed by Pedersen, Patwardhan, Banerjee and Michelizzi (see [15] and [16]). We will not present here word similarity measures only the sentence similarity formula which applies them.

The similarities are defined for concepts. To avoid sense disambiguation for every measure the similarity of two words can be calculated as

$$\mathrm{sim}(w_1, w_2) = \max_{c_1 \in s(w_1), c_2 \in s(w_2)} (\mathrm{sim}(c_1, c_2))$$

where $s(w)$ denotes the different senses of w as concepts. Semantic relatedness is measured only for content-words because "by definition" the content-words are those which bear semantic content.

In our experiments we used the Wu and Palmer measure for word semantic similarity. It is defined as

$$\mathrm{sim}_{\mathrm{WuP}}(c_1, c_2) = \frac{2 \cdot \mathrm{depth}(LCS)}{\mathrm{depth}(c_1) + \mathrm{depth}(c_2)}$$

where LCS is the least or lowest common subsumer of the concepts in the hierarchy. We decided to choose this metric because it bears a resemblance to Lin's measure (actually they are the same if the link strengths are equal in the IS-A hierarchy), which takes into account many characteristics of the semantic network including shortest path between concepts, link strength and concept depth, while the Wu and Palmer measure does not require large corpora to accurately estimate word specificity, however disregards link strength. Experimental and theoretical evaluations of metrics can be found in [15], [1], [19] and [12].

The method described in [4] divides a sentence in open-class word sets, a set created for each content-word type. Then for a sentence for each member of each set the semantically most similar word is found from the second sentence belonging to the set with the same type. Word similarity is measured using one of the above-mentioned metrics, and this value is weighted with the idf (inverse document frequency) of the word:

$$\mathrm{sim}(T_i, T_j)_{T_i} = \frac{\sum_{\mathrm{pos}} \left(\sum_{w \in T_i} (\mathrm{maxSim}(w, T_j) \cdot \mathrm{idf}(w)) \right)}{\sum_{w \in T_i} \mathrm{idf}(w)}$$

The idf which is applied as a word *specificity* factor was introduced to give different weights to words. While giving a greater weight to rare words, it can actually lower the similarity value of two sentences if they contain many common words. Instead of idf we applied ifam (inverse familiarity) which can predict word specificity ([14]), does not

need sense disambiguation and large corpora and is *softer* than idf. Actually one can easily query this in WordNet.

The above sentence-similarity is a one-way measure, this is designated with the index, which means "with respect to T_i". The values obtained from each direction can be combined by taking the average of these.

This measure ignores any relation between words in the sentence, however for example word order is significant when comparing two sentences from different contexts. Thus this measure returns maximal similarity value for the sentences "*The army defeated the enemy*" and "*The enemy defeated the army*" however there is no entailment relation in any direction between these two. The other problem of this measure is that as the length of one sentence increases the resultant value (in one direction) also does. However, with our experiments we tried to show that this metric can be effectively used in text segmentation.

2.2. Clustering

For clustering the segments of documents in a category we used an existing clustering software ([5]). The used algorithm was *k-means* that is still a very good algorithm in very high dimensional spaces. The number of clusters was set to be 10% of the number of segments found in a category, and we used the Euclidean distance as distance metric.

2.3. Support vector machines for text categorization

The basic idea of the supervised learning with SVMs is to find an optimal hyperplane with maximal margin, separating the negative examples from the positive ones. Using maximal margin separating hyperplanes the probability of misclassifying an unseen example is sought to be reduced. The optimization problem turns into a constrained convex quadratic programming task. The Lagrange formulation of this problem will lower the dimensionality of the constraints, thus simplifying the optimization task. But the main advantage of the Lagrange formulation is that allows us to deal with linearly non-separable data in a similar manner as in the linear case. To handle the non-linear case the often high dimensional training data should be mapped to a higher dimensional space to construct the hyperplane there. However, because in this formulation the training vectors appear only in form of dot products, all we need is to give an easily computable function, which returns the value of the dot product (i.e. the value of similarity) of the vectors in that high dimensional space, called feature space. This function is called a kernel function. The simplicity and robustness of the SVMs stands in combining maximal margin hyperplane classification with the above-mentioned "kernel trick" (see for example [2]).

Support vector machines are proven to outperform the other learning techniques applied so far on text categorization task ([8]). Joachims enumerates three main arguments for using SVMs in text categorization:

(i) SVMs can work on large feature spaces

(ii) Document vectors are sparse, therefore efficient decomposition algorithms can be applied

(iii) Most text categorization problems are linearly separable. This was empirically shown by Joachims.

In our experiments we used SVMs to learn the input data vectors constructed using the features extracted by our method. As for a powerful SVM library we used LIBSVM ([3]).

3. Experiments and results

Our experiments were done on the Reuters-21578 corpus, Apte split with 90 categories plus an "unknown" category, which was removed both from the training and the test set.

%	mP	mR	mBEP	mF1	MP	MR	MBEP	MF1
CHI1	87.40	84.69	86.05	86.02	69.40	61.68	65.54	65.31
CHI2	88.46	84.59	86.52	86.48	71.61	61.25	66.43	66.02
SC1	87.29	84.43	85.86	85.83	71.00	61.62	66.31	65.97
SC2	87.97	84.21	86.09	86.04	72.43	60.81	66.62	66.11

Figure 2. Results obtained for the Reuters corpus given in percentage. Notation: mP=micro-precision, mR=micro-recall, mBEP=micro-breakeven, mF1=micro-F_1, MP=macro-precision, MR=macro-recall, MBEP=macro-breakeven, MF1=macro-F_1

The results obtained are shown on Fig.2. CHI1 and CHI2 denote the implemented text categorization systems using the χ^2 feature selection with 6624 and 5209 terms, respectively. In order to be able to compare the systems, the number of the selected features here are the same as the size of the obtained feature set with our method. These two systems use stemming and a stopword list of 199 words. SC1 and SC2 denote our systems with different settings. In SC1 no stemming and stopword removal were performed before the clustering step, but after getting the feature set. In SC2 before clustering the segments we applied stemming and stopword removal.

We used linear SVMs, hence no parameter optimization was needed. As Yang and Liu ([21]) and Joachims ([8]) has already shown the linear mapping, i.e. without mapping the

data to a higher dimensional space, yields good and sometimes better results than for example using the gaussian kernel.

As we already mentioned we used the LIBSVM library for classification and Cluster 3.0 for clustering. The sentence margins were determined using the package Lingua::EN::Sentence, the semantic similarities were calculated using WordNet::Similarity.[1]

It can be seen from the results that our method is a bit worse than χ^2 statistic with the same number of terms considering the microaveraged scores however it has better scores in the macroaveraged case.

4. Conclusions and future work

Our hypothesis is that there are a lot of segments found in the documents belonging to a category that are not directly related to the subject of the category but rather to some feature of it like we have seen in the sample document in which most of the sentences were related to the CIPEC organization that actually deals with the copper market. We also believe that the number of segments of the documents of a category that are referring directly to the subject of the category outnumber any of the indirectly referring segments. So when clustering the segments the largest cluster will contain the words that are used to express direct information about the subject of the category. This hypothesis was shown to be holding by looking at the term sets generated for each category, and be the good performance of the term set used to generate the document vectors.

When comparing our method with term ranking methods we have to emphasize some important dissimilarities:

- our method results in a fixed set of terms that is believed to be optimal and its size does not have to be computed using cross validation. Also, the size of the term set is fairly small, and it depends on the preprocessing of the text (stemming, stop word removal).
- our method guarantees that every category will have a fair amount of terms believed to be relevant to it

The results show that our assumptions work well because a fairly discriminating was found with our method.

As future improvements we plan on increasing the efficiency of text segmentation by either using more complex methods of sentence entailment or other methods. Also we want to further experiment with different clustering methods and feature vector generation of the document segments as sparseness represents a major problem, and also the distance metric is a good target for improvement. There is room for improvement also in testing because our segmentation idea is not yet used for discrimination between the documents' subtopics. Also some ranking method of the terms of the final term set should be devised based on the segment frequency of the terms if the number of the terms is needed to be lowered for example because of the needs of the learning algorithm.

[1] The Perl packages can be found on the CPAN.

Acknowledgements

The research was partly supported by the grant CEEX/1474 of the Romanian Ministry of Education and Research.

References

[1] A. Budanitsky and G. Hirst. Semantic distance in wordnet: an experimental, application-oriented evaluation of five measures. In *Proceedings of the NAACL 2001 Workshop on WordNet and Other Lexical Resources*, Pittsburgh, June 2001.

[2] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. In *Knowledge Discovery and Data Mining*, volume 2, pages 121–167. 1998.

[3] C. Chang and C. Lin. LIBSVM: a library for support vector machines (version 2.31), Sept. 07 2001.

[4] C. Corley, A. Csomai, and R. Mihalcea. Text semantic similarity, with applications. In *Proceedings of International Conference Recent Advances in Natural Language Processing (RANLP 2005)*, September 2005.

[5] M. J. L. de Hoon, S. Imoto, J. Nolan, and S. Miyano. Open source clustering software. *BIOINF: Bioinformatics*, 20:1453–1454, 2004.

[6] J. Furnkranz, T. Mitchell, and E. Riloff. A case study in using linguistic phrases for text categorization on the WWW. In *AAAI/ICML Workshop on Learning for Text Categorization*, 1998.

[7] E. Gabrilovich and S. Markovitch. Feature generation for text categorization using world knowledge. In *Proceedings of The Nineteenth International Joint Conference for Artificial Intelligence*, pages 1048–1053, Edinburgh, Scotland, 2005.

[8] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. Technical Report LS VIII-Report, Universität Dortmund, Dortmund, Germany, 1997.

[9] D. Jurafsky and J. H. Martin. *Speech and language processing*. Prentice Hall, 2000.

[10] H. Kozima. Text segmentation based on similarity. In *ACL*, 1993.

[11] D. D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *SIGIR*, pages 37–50, 1992.

[12] D. Lin. An information-theoretic definition of similarity. In *ICML*, pages 296–304, 1998.

[13] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, February 2002.

[14] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Five papers on wordnet. Technical Report, Cognitive Science Laboratory, Princeton University, 1993.

[15] S. Patwardhan, S. Banerjee, and T. Pedersen. Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, 2003.

[16] T. Pedersen, S. Patwardhan, and J. Michelizzi. Wordnet:: Similarity - measuring the relatedness of concepts. In *AAAI*, pages 1024–1025, 2004.

[17] C. J. V. Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[18] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[19] N. Seco, T. Veale, and J. Hayes. An intrinsic information content metric for semantic similarity in wordnet. In *ECAI*, pages 1089–1090, 2004.

[20] G. Varelas. Semantic similarity methods in wordnet and their application to information retrieval on the web. Technical Report TR-TUC-ISL-01-2005, Technical Univ. of Crete (TUC), Dept. of Electronic and Computer Engineering, Chania, Crete, Greece, 2005.

[21] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR '99*, pages 42–49, 1999.

[22] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *ICML*, pages 412–420, 1997.