Journal of ELECTRICAL ENGINEERING, VOL. 56, NO. 12/s, 2005, 94–97

A SIMILARITY–BASED APPROACH IN TERM WEIGHTING FOR TEXT CATEGORIZATION

Vladimír Repiský — Martin Homola *

This paper presents a new approach to term weighting in the machine learning text categorization and describes a preliminary implementation. Comparisons with the standard approaches are promising.

K e y w o r d s: text categorization, machine learning, training set

2000 *Mathematics Subject Classification*: 94A05

1 INTRODUCTION

Automated text categorization is a task to assign one or more symbolic categories to a natural language document. In recent years, this task has been extensively addressed by machine learning (see [6] for a survey). In order to utilize machine learning in text categorization, a training set of preclassified documents is necessary. The classifier is build by analyzing the documents of the training set. Also, feeding natural language documents directly into the machine learning process is not feasible. Hence, these documents are preprocessed. A compact representation is assigned to each document. The document's lexical structure (e.g. order of words, etc.) is usually ignored at this point. It is our stance that taking the lexical structure into account may improve the results of the classification. Some approaches that work with the lexical structure of documents are already known (such as [1]).

We introduce a new algorithm to compute term weights for machine learning based text categorization. Our approach uses a language model based on the InfoMap software from [7] that allows us to compute the semantic closeness of words. This model has been successfully applied in information retrieval, clustering and word sense disambiguation [7,5]. The performance of our weighting functions is comparable to the state-of-the-art approaches (see [3] for a survey on term weighting for text categorization).

2 AUTOMATED TEXT CATEGORIZATION

Let $\mathcal{D}$ be a set of documents. Let $\mathcal{C}$ be a set of categories. The categories are just symbolic labels. The problem of text categorization is to assign the labels of $\mathcal{C}$ to the documents of $\mathcal{D}$. Zero or more categories may be assigned to each document and also one category may be assigned to zero or more documents, it is a many-to-many relationship. More formally, there is a function $\breve{\Phi}\colon \mathcal{D} \times \mathcal{C} \to \{\text{true}, \text{false}\}$ called the *ideal classifier* that tells us how to do it, however, we do not know this function, at least not for each pair of $d \in \mathcal{D}$ and $c \in \mathcal{C}$. And, the task is to find the best possible approximation of $\breve{\Phi}$, the *classifier* function $\Phi\colon \mathcal{D} \times \mathcal{C} \to \{\text{true}, \text{false}\}$.

If there is only one category $c \in \mathcal{C}$ and its complementary category $\overline{c} \in \mathcal{C}$ then such a text categorization task is called *binary*. In other words, the task is only to determine whether a document classifies in the category c or not. According to [6], the general, also called *n-ary* case, is reducible to the binary case: it is sufficient to construct one binary classifier $\Phi_{c_i} : \mathcal{D} \to \{\text{true}, \text{false}\}$ for each category $c_i \in \mathcal{C}$. Hence we only deal with binary classifiers here on.

It is not feasible for most machine learning techniques to process natural language documents. Therefore a compact representation is assigned to each document for the learning algorithm to process. The lexical structure of documents is usually ignored at this point. Documents are parsed into *terms* (also *features*). The significance of each term for a particular document is measured somehow (this is called *term weighting*). Only the most significant terms are selected (this is called *term selection*). Then, the document is represented by a vector of weights of the selected terms. Term weighting and the dimensionality reduction by term selection play a crucial role in the text categorization and have a great impact on the overall performance (cf. [6,3]). We review the commonly used term weighting functions in Section 3.

To address the text categorization task by means of machine learning, some partial knowledge of the ideal classifier $\breve{\Phi}$ is needed. The set of documents $\{d_1, \ldots, d_n\} \subseteq \mathcal{D}$, for which the values of $\breve{\Phi}$ are known, is denoted by Ω. The classifier is constructed by inspecting the documents of Ω and the associated values of $\breve{\Phi}$. In machine learning, this phase is commonly called *training*. Then the classifier is used to classify the remaining documents of $\mathcal{D}$. For the purpose of research, the performance of classifiers has to

* Department of applied informatics, Faculty of Mathematics Physics and Informatics, Comenius University, Bratislava, Slovakia, E-mail: vladimir.repisky@st.fmph.uniba.sk, homola@fmph.uniba.sk

ISSN 1335-3632

Table 1. Popular term weighting functions.

Function	Definition
binary	1 if $t_i \in d$ and 0 otherwise
tf	occurrence count of t_i in d
tf·rf	$\mathrm{tf} \cdot \log\left(1 + \frac{n_i}{n_i-}\right)$
log tf	$1 + \log(\mathrm{tf})$
itf	$1 - \frac{r}{r-\mathrm{tf}}$, usually $r = 1$

Table 2. The five most similar word-stems for each category of the ten largest Reuters-21578 categories.

Cat.	Five most similar word-stems
earn	receiv, scholarship, career, covet, salari
acquisit	acquir, sharehold, corpor, ventur, mutual
money	cash, pay, payment, buy, repay
grain	wheat, lumber, crop, corn, timber
crude	petroleum, oil, copra, export, refineri
trade	trader, tariff, gatt, protectionist, barter
interest	investor, debt, money, lender, loan
ship	vessel, warship, frigat, battleship, merchant
wheat	barley, maiz, beet, dairi, veget
corn	wheat, potato, maiz, barley, cereal

be evaluated. It is critical to do the evaluation on documents unseen during the training phase and hence Ω is split into the *training set* $\mathrm{Tr} \subseteq \Omega$ and the *testing set* $\mathrm{Te} \subseteq \Omega$ such that $\mathrm{Tr} \cap \mathrm{Te} = \emptyset$. The training is done only on the documents of Tr and the evaluation only on the documents of Te.

It seems that almost any machine learning algorithm designed for supervised learning is applicable on text categorization (cf. [6]). We have selected the state-of-the-art SVM algorithm. We have used polynomial kernel function for SVM. The polynomial kernel function performs better than other kernel functions. We have used the SVM^{light} 6.01 [2] software for SVM.

3 TERM WEIGHTING FUNCTIONS

A number of term weighting functions has been used in text categorization. A comparative survey is found in [3]. We summarize the most interesting functions from [3] in Table 1, in which we show how to compute the weight of a term t_i in a document d, while classifying into a category c. Here, N is the total number of documents in Tr, n_i is the number of documents in Tr that contain t_i, and n_i- is the number of documents in Tr that contain t_i and do not belong to c.

According to [3], the raw term frequency tf outperforms binary. They have used the rf factor and showed that the resulting tf·rf function outperforms tf. Also the inverse term frequency itf yields interesting results. However, the difference is rather small, and hence, the actual best choice for a particular application may be context dependend. For more details the reader is referred to [3].

4 TERM WEIGHTING BASED ON SEMANTIC CLOSENESS

It is our stance, that the popular weighting functions such as tf may be outperformed if the lexical structure of documents is taken into account. The meaning of a document is not determined just by the words that are comprised in the document but also by the way how these words are combined to form the document. Results of [7,5] in information retrieval and word sense disambiguation encourage us in such consideration.

4.1 Language Model

The weight we assign to a term t in a document is computed from the *semantic closeness* of t with the other terms in the document. Selecting words for terms is a natural choice here. In order to measure the semantic closeness of words we need a language model. We use the InfoMap software[1] of [7] on a corpus of documents to produce this model. The InfoMap algorithm is based on term cooccurrence. Arbitrary corpus may be processed by InfoMap, we have chosen the corpus of documents from Wikipedia[2]. Porter's stemming algorithm [4] was used to unify words with same stem.[3] We will now summarize how the semantic closeness is computed using the InfoMap software.

We first select some interesting terms from the the corpus. Stop words and also terms that rarely occur are removed. From the remaining m terms, the n most frequent terms are selected as *content bearing* ($m = 100,000$ and $n = 2,000$ in our experiment). For these terms the term-cooccurrence matrix $A = \|a\|_{m \times n}$ is computed. The element $a_{i,j}$ is the cooccurrence count of the i-th term with the j-th content bearing term within a window of a certain size (15 in our experiment) summed together for all documents of the corpus. The semantic closeness of two terms t_i and t_j is determined by the cosine similarity measure of the i-th and the j-th row-vector of A.

For the sake of optimization the matrix A is further processed. First, partial singular value decomposition with rank k (we have used $k = 500$) is applied to A: we find a diagonal matrix $S = \|s\|_{k \times k}$ and two matrices $U = \|u\|_{m \times k}$, $V = \|v\|_{n \times k}$ such that $A' = USV^\top$ is

[1] InfoMap-nlp software is available at: http://infomap-nlp.sourceforge.net/ .

[2] Wikipedia – the free encyclopedia contains over 500,000 articles in English. A database-dump of Wikipedia is available at: http://download.wikipedia.org/ . For easier parsing, we have converted this dump using the WikiToTome software that is freely available at: http://members.chello.nl/epzachte/Wikipedia/ProcedureTR3.html

[3] We have used implementation of Porter's stemming algorithm from https://sourceforge.net/projects/porterstemmers/ .

an approximation of A. Then, the rows of A' are normalized so that for each row i the equation $\sqrt{\sum_j {a'_{i,j}}^2} = 1$ holds. Now an approximation of the cosine similarity of the i-th and the j-th term is simply a dot-product of the i-th and the j-th row of A', that is the i,j-th element of the matrix $C = A'A'^{\top}$. All this is computed using the InfoMap software, reader is referred to [7] for more details. An illustration of word-stems that are semantically close according to this model is found in Table 2.

4.2 Derived Weighting Functions

In our approach, the weight assigned to a term in a document is given by the term's activity in the document. A given term t_i is cumulatively activated by all the other terms in the document that are semantically close to t_i. Formally, the *activity* of a term t_i *caused by* a term t_j is simply the semantic closeness between t_i and t_j, that is the i,j-th element $c_{i,j}$ of the matrix C. The *activity* of a term t_i *in a document* d is then

$$\mathrm{A}^d_{t_i} = \sum_{t_j \in d \ \wedge \ c_{i,j} > \theta_A} c_{i,j} \ . \qquad (1)$$

The reason for using the threshold θ_A is to use only the semantically close terms for the activation and to suppress activation by unrelated terms. We have tried different values for θ_A, see Section 5 for discussion. The activity of a term in a document itself serves as a weighting function. Since we have used term cooccurrence in our approach, we call this function *cooccurrence-based activity weight* and denote it as caw. Formally,

$$\mathrm{caw}(t,d) = \mathrm{A}^d_t \ .$$

In our experiments we have also tried $\log\mathrm{caw} = 1 + \log(\mathrm{caw})$, $\mathrm{icaw} = 1 - \frac{1}{1-\mathrm{caw}}$ and $\mathrm{caw}\cdot\mathrm{rf} = \mathrm{caw} \cdot \log\left(1 + \frac{n_i}{n_{i-}}\right)$.

We use these weighting functions to preprocess the documents and then we feed the computed weights into the SVM classifier. This enables us to compare them with other successful weighting functions such as tf, log tf and tf·rf. We summarize our results in the following section.

5 RESULTS

For benchmarking we have used the Reuters-21578 corpus. It is a collection of news stories preclassified by human indexers. We have followed the ModApte split in which 12,092 stories are split into the training set (75%, 9,603 stories) and the testing set (25%, 3,299 stories). We have used the restriction ModApte[10], which contains only the 10 categories with the highest number of the positive training documents.

From the documents we have removed stopwords and words shorter than three letters. After stemming the total count of different words was 27,093. From the Wikipedia corpus we have computed the semantic closeness for 100,000 words, from which 15,109 are among these 27,093 words. Our algorithm is working only with these 15,109 words. We have performed tests with standard weighting functions, which select features for categorization both from 15,109 and from all 27,093 words.

We have tried different threshold values in (1). The performance of caw for different threshold values is depicted in Fig. 1. The best results are for the threshold value $\theta_A = 0.5$, hence this value is used in all later tests.

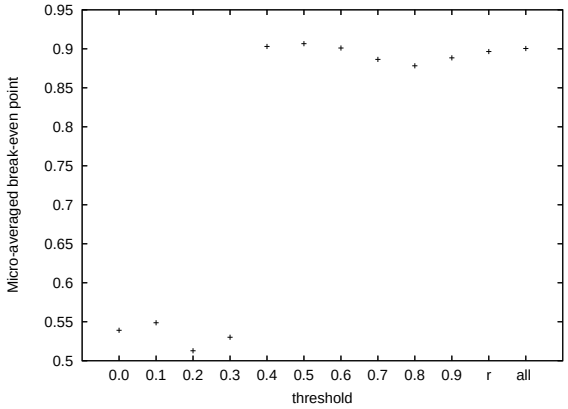

Fig. 1. The comparison of caw for different threshold values, with tf for full (labeled "all") and for restricted number of words (labeled "r") on Reuters-21578 in the ModApte[10] split. Feature size is 3,000.

A common performance measure in text categorization is the break-even point (see [6]). We have measured the micro-averaged break-even points for 5 different feature sizes: 1,000; 2,000; 3,000; 5,000 and 8,000. We have used the GSS_{max} (see [6]) function for feature selection. The comparisons of our algorithm with standard weighting functions on restricted and full word set are depicted in Figs. 2 and 3. Especially caw·rf performs very well. In both cases it is comparable to the state-of-the-art tf·rf function.

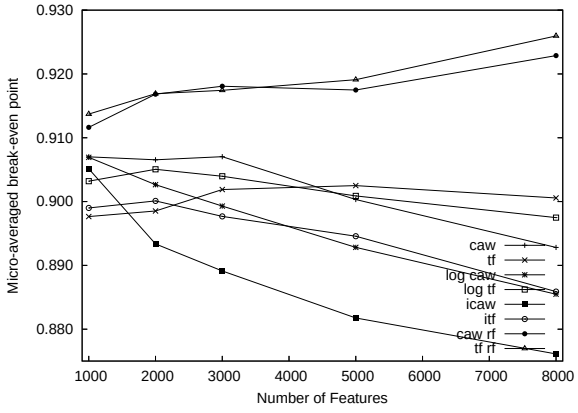

Fig. 2. The comparison of our algorithm with standard weighting functions on the restricted word set.

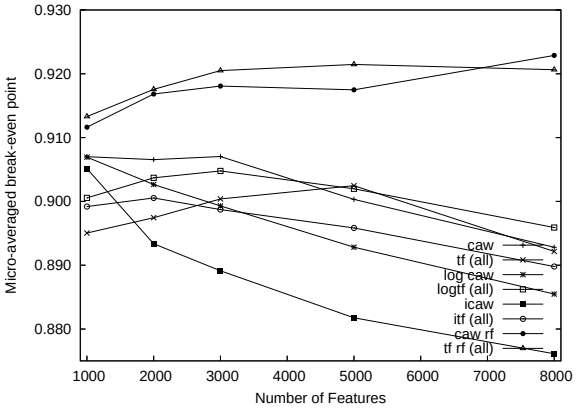

Fig. 3. The comparison of our algorithm with standard weighting functions on the full word set.

6 CONCLUSION

We have introduced a new algorithm to compute term weights in machine learning text categorization. Our approach uses a language model based on the InfoMap software from [7] that allows us to compute the semantic closeness of words. This model has been previously used in information retrieval and word sense disambiguation. We have made a test implementation, however we were not able to compute the semantic closeness for a sufficient number of words so far. The main drawback of our approach is in high computational requirements. Our weighting functions performance is comparable to the state-of-the-art weighting functions described in [3]. In our ongoing experiments we will try to compute the semantic closeness for a higher number of words to see, if the performance can be improved.

Acknowledgments

Authors would like to thank to Blažej Dolista for his help with the implementation. Authors acknowledge financial support from the Slovak scientific grant agency VEGA grant no. 1/0173/03 and grant no. 314/2005 awarded by Comenius University.

References

[1] BERGSMA, S.: Title Similarity-Based Feature Weighting for Text Categorization, Tech. Report CMPUT 650, Department of Computer Science, University of Alberta, 2004.

[2] JOACHIMS, T.: Making Large-Scale Support Vector Machine Learning Practical, Advances in Kernel Methods – Support Vector Learning, MIT-Press, 1999.

[3] LAN, M.—SUNG, S.-Y.—LOW, H. B.—TAN, C.-L.: A Comparative Study on Term Weighting Schemes for Text Categorization, Procs. of IJCNN'05, 2005.

[4] PORTER, M. F.: An Algorithm for Suffix Stripping, Program **14**(3) (1980), 130–137.

[5] SCHÜTZE, H.: Automatic Word Sense Discrimination, Tech. report, Association for Computational Linguistics, 1998.

[6] SEBASTIANI, F.: Machine Learning in Automated Text Categorization, ACM Comuting Surveys **34** No. 1 (2002).

[7] TAKAYAMA, Y.—FLOURNOY, R. S.—KAUFMANN, S.: Information Mapping: Concept-Based Information Retrieval Based on Word Associations, available online: http://www-csli.stanford.edu/semlab/infomap/Papers/takayama-infomap.ps.

Received 1 June 2005

Vladimír Repiský is a student of computer science at the Faculty of Mathematics Physics and Informatics of the Comenius University in Bratislava, Slovakia. His interests include machine learning and neural networks.

Martin Homola is a graduate student of artificial intelligence at the Faculty of Mathematics Physics and Informatics of the Comenius University in Bratislava, Slovakia. His interests include knowledge representation and overall artificial intelligence.