Reducing the dimensionality of bag-of-words text representation used by learning algorithms

Article · January 2003

CITATIONS
18

READS
789

2 authors:

Claudia Martins
Federal University of Mato Grosso
20 PUBLICATIONS **67** CITATIONS
SEE PROFILE

Edson Takashi Matsubara
Federal University of Mato Grosso do Sul
73 PUBLICATIONS **1,581** CITATIONS
SEE PROFILE

Reducing the Dimensionality of Bag-of-Words Text Representation Used by Learning Algorithms

Claudia Aparecida Martins* Maria Carolina Monard Edson Takashi Matsubara

Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação e Estatística
13560-970, São Carlos, SP, Brazil
e-mail: {cam, mcmonard, edsontm}@icmc.usp.br
* On leave from Universidade Federal de Mato Grosso, Brazil

ABSTRACT
The attribute-value representation of documents used in Text Mining provides a natural framework for classifying or clustering documents based on their content. Supervised learning algorithms can be applied whenever the documents have labels preassigned or unsupervised learning for unlabeled documents. The attribute-value representation of documents is characterized by very high dimensional data since every word in the document may be treated as an attribute. However, the representation of documents has a crucial influence on how well some supervised learning algorithm can generalize. This work presents a way to efficiently decomposing text into words (stems) using the bag-of-words approach as well as reducing the dimensionality of its representation, making text accessible to most Machine Learning algorithms that require each example be described by a vector of fixed dimensionality. A computational tool we have implemented is used on a real case in order to illustrate our proposal as well as several of the facilities implemented in the tool which allow to improve the accuracy of classifiers through the reduction of the dimensionality of text representation.

KEY WORDS
Text Mining, Machine Learning, Inductive Learning.

1 Introduction

The task of automatically assigning categories to natural language text has become one of the key methods for organizing online information. Since hand-coding such classification rules is costly or even impractical, most modern approaches employ Machine Learning techniques to automatically learn text classifiers from examples.

However, it is necessary to transform texts, or documents, into a form appropriate to interpretation by a classifier-building algorithm. These issues include term weighting and dimensionality reduction. The representation of documents has a crucial influence on how well the learning algorithm can generalize (Sebastiani, 2002).

The objective of this work is to describe a way to reduce the dimensionality of the attribute-value table, that represents a collection of documents, considering the precision of classifiers induced by learning algorithms. Our proposal is illustrated on a real case using a text preprocessing tool we have implemented, called PRETEXT.

This work is organized as follows: Section 2 briefly describes the phases of the Text Mining process; Section 3 presents an overview of text preprocessing based on *bag-of-words* approach used in this work; Section 4 describes the main characteristics of PRETEXT and the Section 5 the experiments. Finally, Section 6 presents the conclusions.

2 The Text Mining Process

A collection of documents $D = \{d_1, d_2, \ldots d_n\}$ and a set of categories $C = \{c_1, c_2, \ldots c_z\}$ associated with the collection of documents D are required in order to perform a Text Mining — TM — task. In this work we shall concentrate in the categorization task, using a simple representation of the documents, which consists of inducting a classifier with the aim of determining if a document d_i belongs to a category c_j with $i = 1, 2, \ldots n$ and $j = 1, 2, \ldots z$.

Besides the categorization task, other tasks such as document summarization and clustering may be related to the TM process, depending on the application nature and if the set of categories C is known. In any case, some phases are essential to the TM process. These phases are (1) document collection; (2) text preprocessing; (3) knowledge extraction; (4) results evaluation and interpretation.

After the document collection phase, the collected documents should be transformed to a format accepted by the knowledge extraction algorithms. This second phase, text preprocessing, outputs a structure, frequently represented as an attribute-value table, that represents the document collection. This phase is computationally intensive and careful text preprocessing is essential to the success of the TM process. More details are presented in Section 3.

As soon as the documents are represented in a convenient format, algorithms for extracting knowledge can be

applied with the objective of discovering patterns in the documents that are useful and previously unknown. Finally, the evaluation phase verifies if the objectives have been reached or if any of the phases should be carried out again.

3 The Text Preprocessing Phase

During this phase, the collected documents should be preprocessed in order to transform their representation. Frequently, each document d_i is transformed to a vector of terms that occur in the document. The identification of the terms in a document may be based on the words present in the text (bag-of-words), or more sophisticated representations (Montes-y-Gómez, Gelbukh, López-López, & Baeza-Yates, 2001).

However, results obtained by experimental research works have shown that more sophisticated representations could present inferior performance as compared with representations based on simple words (Apté, Damerau, & Weiss, 1994; Dumais, Platt, Heckerman, & Sahami, 1998; Lewis, 1992). According to Lewis (1992), the most probable explanation of these results is that, even if more sophisticated terms present a superior semantical quality, the statistical quality is inferior as compared with simple word terms. In this way, research works related to the use of simple and sophisticated representations of documents are active (Montes-y-Gómez, Gelbukh, & López-López, 2002).

In the identification of terms as bag-of-words, a term can be represented by simple words (1-gram) or composed words ($2, 3, \ldots, n$-gram) that occur in the document. Each term is used as an attribute of the data set represented in the attribute-value form. The set of documents, after preprocessing, can be represented as shown in Table 1.

Table 1. Representation of documents

	t_1	t_2	...	t_m	C
d_1	a_{11}	a_{12}	...	a_{1m}	c_1
d_2	a_{21}	a_{22}	...	a_{2m}	c_2
...	...	...	...	...	...
d_n	a_{n1}	a_{n2}	...	a_{nm}	c_n

In Table 1 n documents (examples) are represented, and each document is composed by m terms. Each document d_i can be understood as a vector $d_i = (a_{i1}, a_{i2}, \ldots a_{im})$. An a_{ij} value refers to the value of the jth term of the ith document, i.e., a_{ij} represents the value of the term t_j in the document d_i. If binary values are used, the value 1 means the presence of the term t_j in the document d_i, and the value 0 means the term is absent. However, the binary representation of terms might be inadequate for some applications. Thus, statistical measures that take into account the frequency a term appears in a document and the frequency this term is found in other documents might provide better results. For instance, *term frequency* — tf — is a measure that counts the number of occurrences of a term t_i in a document d_i, and is defined by $tf(t_i, d_i) = \#(t_i, d_i)$, where $\#(t_i, d_i)$ is the number of occurrences of t_i in d_i.

Terms with high frequency, i.e., terms that occur in all (or in the majority) of the documents, frequently do not present useful information to discriminate the documents. The measure *inverse document frequency* — idf — favors terms that occur in few documents of the collection. idf is inversely proportional to the number of documents having a certain term in a document collection, and is defined by $idf = \log \frac{n}{\#Dt_j}$, where n is the total number of documents in the document collection D, and $\#Dt_j$ is the number of documents in D where the term t_j occurs at least once.

Another relevant aspect that should be considered when the measures tf and idf are applied is related to documents that have a large difference in the number of terms. Since larger documents usually have a higher probability of being relevant than smaller documents. However, all relevant documents should be considered as equally important, independently of their size. In this case, a normalization factor should be incorporated with the aim of normalizing the a_{ij} values of documents. The $tfidf$ measure is extended to incorporate the normalization factor and is defined by $tfidfn(t_j, d_i) = \frac{tfidf(t_j, d_i)}{\sqrt{\sum_{s=1}^{n} (tfidf(t_s, d_i))^2}}$.

The presented measures are commonly used to assign values to terms that occur in the documents. However, in an attribute-value table, each attribute is a term that occurs in the documents. Thus, a criteria to reduce the dimensionality of the data set should be considered. Several methods can be used in order to reduce the amount of attributes in the data set. The reduction of attributes can result in more representative attributes and in a better performance of the TM process. A widely used method consists of reducing each term to its radical. This transformation can be accomplished by the use of *stemming* algorithms, which are widely used in text processing.

Basically, the algorithm consists of a linguistic normalization, in which the variant forms of a term are reduced to a common form, called *stem*. The stemming algorithm is strongly dependent on the document language. A well known stemming algorithm is the Porter's algorithm that removes suffixes of English terms (Porter, 1980). This algorithm has been widely used, cited and adapted in the last 20 years. Several implementations are available in the Word Wide Web, among them there exists an implementation maintained by the algorithm author (http://www.tartarus.org/ martin/PorterStemmer).

Another approach to reduce the dimensionality of the data set is to select the most representative terms that discriminate the documents. The *Zipf's law* (Zipf, 1949) describes an approach to find out which terms are likely to be unrepresentative in a document collection. The Zipf's law for documents can be applied not only to terms, but also to phrases and sentences. Actually, the Zipf's law is an empirical observation that can be applied to several domains, and follows the distribution $p_1 = \frac{c}{1}, p_2 = \frac{c}{2}, \ldots, p_n = \frac{c}{n}$,

where $c = \frac{1}{H_n}$ and $H_n = \sum_{i=1}^{n} \frac{1}{i}$ (Knuth, 1973).

There are several ways to enunciate the Zipf's law for a collection of documents. The most simple one is procedural: to collect all terms from a document collection and to count the number of times that each term occurs in the documents. If the resulting histogram is disposed in a decreasing order, i.e., the most frequent term appears first, then the resulting curve is called *Zipf curve*. If the Zipf curve is plotted in logarithmic scale, then the curve looks like a straight line with slope = -1, although this seems to depend on language and style, as shown in (Gelbukh & Sidorov, 2001).

Luhn (1958) used this law as a null hypothesis in order to specify an upper and a lower cut-offs, with the aim of excluding irrelevant terms. The terms that exceed the upper cut-off are the most frequent terms. These terms are common in all documents and are usually prepositions, conjunctions and articles. On the other hand, the terms below the lower cut-off can be considered too rare and, consequently, they do not contribute significantly in the document discrimination. Thus, Luhn proposed an approach to select the most relevant terms, considering that the most significant terms are positioned between the two cut-offs. However, the exact determination of the cut-offs is frequently obtained in a trial and error process (Van Rijsbergen, 1979). The following section presents a computational tool that uses the presented concepts.

4 The PRETEXT Tool

PRETEXT (Matsubara, Martins, & Monard, 2003) is a computational tool implemented in Perl (Wall, Christiansen, & Schwartz, 1996) using the object oriented paradigm. The tool was developed with the aim of automatically performing the text preprocessing tasks on a collection of documents. The documents may be written in three different languages: Portuguese, Spanish and English. The tool implementation is based on the Porter's stemming algorithm for the English language, which was adapted to the Portuguese and Spanish languages. In addition, the tool includes facilities to reduce the dimensionality of the data sets using the Zipf's law and the Luhn cut-offs.

PRETEXT supports several facilities supported by similar tools, such as the Ngram Statistics Package (Banerjee & Pedersen, 2003) as well some other facilities which will allow us to integrate PRETEXT into the DISCOVER environment described latter.

In a general way, the main preprocessing tasks performed by the PRETEXT tool are: (i) to extract stems of words in Portuguese, Spanish and English; (ii) to ignore words that can be considered non-significative through the use of a *stopword* list; (iii) to create intermediate files that store the stem frequencies in each document, the stem frequencies in the entire document collection and the frequency of the words which were used to create each of the stems; (iv) to use any of the four measures presented in Section 3 to assign a value to each stem in a document collection; (v) to apply the Zipf's law and Luhn cut-offs; (vi) to deal with simple or composed terms — 1, 2 and 3-gram; (vii) to plot stem frequencies and other relevant graphics; (viii) to create an attribute-value table using values assigned to selected stems.

PRETEXT uses a standard stopword list for each language. This list has general terms such as articles, conjunctions, prepositions, pronouns and some adverbs. The standard stopword list is stored in a file and the user can use, in addition to the standard list, other lists with stopwords that are specific to the application domain. The tool was designed to use a set of stopword files. In order to automatically perform the Luhn cut-offs, PRETEXT has an option to consider only stems in the frequency range $(\bar{x}-ks; \bar{x}+ks)$, in which $\bar{x}$ is the mean of the stem frequencies, s is the standard deviation and k is a constant defined by the user. Another option allows the user to define the Luhn upper and lower cut-offs.

The tool, illustrated in Figure 1, consists of two main modules: **Stem.pl** and **Report.pl**. The first module is responsible for transforming words into stems. The input for this module can be a word, a document or a document collection. Figure 1 shows the third case, in which a collection of documents is identified by the name of the directory that stores a set of files, and each file stores a document. The user should inform the document collection language, i.e., Portuguese, Spanish or English and, if necessary, a user defined stopword list, which will be added to the standard stopword list of PRETEXT. Also, the tool creates several intermediate files (item (iii)).

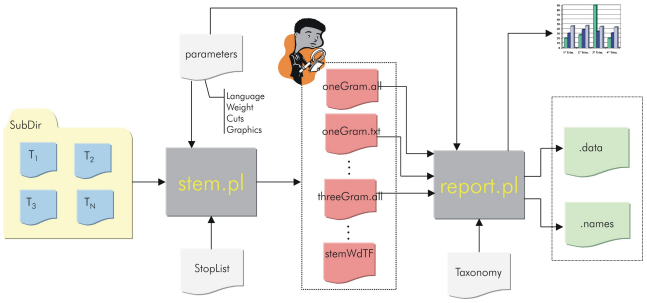

Figure 1. The PRETEXT tool

The **Report.pl** module uses as input some of the intermediate files generated by the **Stem.pl** module, and an additional file where the execution parameters are specified. In this additional file should be defined parameters such as the measure to assign values to stems, the upper and lower Luhn cut-offs, as well as the 1, 2 or 3-gram to be considered (simple or composed terms). The default tool values are: the tf measure, no cuts (all stems are considered) and 1, 2 and 3-gram (simple and composed stems). The **Report.pl** module outputs a `.data` and `.names` files in the DISCOVER[1] standard syntax. In addition, there are

[1]The PRETEXT tool will be integrated into the DISCOVER environment which is a research project under development in the Laboratory

several graphics facilities to show the stem frequencies in the document collection and others.

5 A Case Study

In order to illustrate our proposal, results from some experiments using PRETEXT on a corpus of documents obtained from the Interinstitutional Center for Research and Development in Computational Linguistics — NILC[2] — are shown. This corpus contains more than 4000 documents in Brazilian Portuguese divided in the following topics: didactic, journalistic, juridical, literary and technic/cientific. We selected 248 documents from the journalistic topic, which are classified into four classes: Informatics, Economy, Sport and Politics. Each document is a text file (`txt` extension) having a mean size and standard deviation of 24.84 KB±13.76 KB. Table 2 shows the document distribution by classes.

Table 2. Number of documents in the classes

Informatics	Economy	Sport	Politics	Total
66 (26.61%)	63 (25.40%)	59 (23.79%)	60 (24.20%)	248

The experiments using this corpus were done to illustrate some of the steps generally used to reduce the dimensionality attribute space, using PRETEXT, and to induce a good classifier, using learning algorithms. In this work, we used the $\mathcal{S}$ee5 (Rulequest-Research, 1999), $\mathcal{CN}2$ (Clark & Boswell, 1989) and SVMTorch II (Collobert & Bengio, 2001) learning algorithms. $\mathcal{S}$ee5 and $\mathcal{CN}2$ are symbolic learning algorithms that induce decision rules where each rule describes a specific context associated with a class. Nevertheless, the inductive bias of $\mathcal{CN}2$ is quite different than $\mathcal{S}$ee5 (Baranauskas & Monard, 2000).

Support Vector Machines — SVM — are learning techniques based on the Statistical Learning Theory, proposed by Vapnik & Chervonenkis (1971). They map the input data to an abstract space of high dimension, where the examples can be efficiently separated by an hyperplane. The SVM incorporates this concept with the use of functions named *Kernels*. These functions allow the access to complex spaces in a simplified and computationally efficient way. The optimal hyperplane in this space is defined as the one that maximizes the separation margin between data belonging to different classes. The main advantages of SVMs are their precision and their robustness with high dimensional data. However, unlike decision trees, SVM classifiers are not directly interpretable.

First of all, the documents were processed with the PRETEXT tool without specifying Luhn cut-offs. There were 22214 stems where more than 9000 stems appeared only once and more than 2900 appeared twice. Thus, we decided to keep only stems above a given frequency threshold. This threshold was defined considering the frequency above 10% of the minority class, i.e., stems with frequency above 6 for this document collection. In this case, only 6202 stems remained where the average frequency of stems is 392.4±415.1. Figure 2 shows one of the graphic outputs from PRETEXT. It shows that there is one stem that appears 3217 times and there are only stems that appear with frequency equal or greater than 6 in the collection of documents.

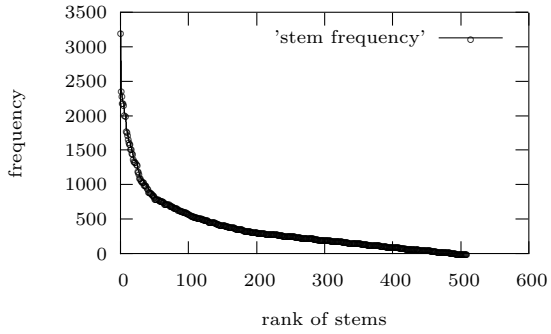

Figure 2. Frequency distribution of stems

The next step is to determine the Luhn cut-offs and to execute PRETEXT creating the attribute-value table in the DISCOVER format (`.data` and `.names` files). Then, using this table it is possible to observe the error rate of classifiers induced by some learning algorithm and, if necessary, to adjust the Luhn cut-offs and to run the algorithms again.

The results obtained by the two symbolic learning algorithms are presented in Table 3, where 'Exp' identifies the experiment; 'm' and 'M' are respectively the lower and upper values used as Luhn cut-offs; 'Att' is the number of attributes (stems) in the attribute-value table; 'Ms' identifies the measure used in the experiment, tf or $tfidfn$ (represented by tn); '10-fold_error %' is the percentage of the error of the classifier induced by $\mathcal{S}$ee5 (S) and $\mathcal{CN}2$ (C), using 10-fold cross validation; 'Rules' is the number of rules in the rule set induced by the algorithms, and 'Att_R' is the number of different attributes present in the rule set.

The symbol $_*$ in experiments 3 and 5 indicates that the number of attributes was reduced not only by using the Luhn cut-offs but afterwards they were reduced further using the information contained in some of the files created by the **Stem.pl** module. With these information the user can decide if a stem represents a set of relevant words or if some of the words having identical stems are meaningful to discriminate documents. Words that are not considered relevant by the user can be included in the user stopword list and PRETEXT can be run again, thus reducing further the data dimensionality.

In the first experiment, the attribute-value table with stems with frequency above 6 was submitted to the $\mathcal{S}$ee5

of Computational Intelligence — LABIC —`http://labic.icmc.usp.br`. The DISCOVER environment aids in the planning and execution of experiments related to the application of learning systems in the Data and Text Mining process (Batista, 2003; Prati, 2003).

[2]Núcleo Interinstitucional de Lingüística Computacional - http://www.nilc.icmc.usp.br/nilc/

Table 3. $\mathcal{S}$ee5 and $\mathcal{CN}2$ experimental results

Exp	m	M	Att	Ms	10-fold_error %		Rules		Att_R	
					S	C	S	C	S	C
				tf	8.8 ± 2.1	8.5 ± 1.8	6	13	7	21
1	6		6202	tn	8.8 ± 1.5	8.5 ± 1.9	6	13	7	21
				tf	8.4 ± 2.7	7.7 ± 1.9	6	14	7	20
2	6	808	6142	tn	6.9 ± 2.0	7.2 ± 2.4	6	14	7	20
				tf	9.3 ± 2.2	10.1 ± 1.9	7	12	8	23
3_*	59		1538	tn	10.1 ± 2.1	10.9 ± 1.8	7	12	8	23
				tf	7.7 ± 1.7	12.1 ± 2.1	6	13	7	21
4	30		2468	tn	7.7 ± 1.0	10.9 ± 1.6	6	13	7	21
				tf	7.6 ± 1.5	8.5 ± 2.3	6	13	6	22
5_*	30	808	2412	tn	8.4 ± 1.8	7.6 ± 1.5	6	13	6	22
				tf	8.4 ± 1.7	10.5 ± 1.9	6	13	7	21
6	12		4266	tn	8.5 ± 2.0	10.5 ± 2.3	6	13	7	21
				tf	6.9 ± 1.1	9.3 ± 1.8	6	13	7	21
7	24		2874	tn	8.4 ± 1.5	7.3 ± 0.8	6	13	7	21
				tf	7.7 ± 1.5	12.9 ± 2.1	7	12	8	23
8	48		1796	tn	8.5 ± 1.6	9.7 ± 2.3	7	12	8	23
				tf	2.8 ± 1.2	2.9 ± 1.4	7	12	6	15
9	6	808	19	tn	4.0 ± 1.2	3.6 ± 1.1	7	12	6	15

and $\mathcal{CN}2$ algorithms. The error rate of both classifiers can be considered similar except for the number of rules induced. As compared with the majority class error, the errors can be considered good.

The idea used in experiment 2 is to find the appropriate M cut-off value. The value 808 was chosen because it is related to the mean of stem frequencies and one standard deviation (392.4±415.1). In this case, only attributes having frequency less than 6 and greater than 808 were discarded. The results obtained show an improvement specially for the classifier induced by $\mathcal{S}$ee5 using the $tfidfn$ measure.

In order to use an attribute value that can discriminate perfectly a class if exists, in the experiment 3_* the stems were used with frequency equal or greater than 59. This value represents the number of examples that belong to the minority class which is given by class Sport — Table 2. This can be considered a very rough heuristic which is based on the idea that one attribute may appear only once in all documents of the same class, being an attribute which perfectly discriminates that class. Thus, the safer value is given by the number of documents in the minority class. However, the error of classifiers increased using this value.

In order to continue the trial and error process to find a good lower Luhn cut-off value, experiments 4 and 5_*, 6, 7 and 8 were run setting this value as approximately 50%, 20%, 40% and 80% of the number of examples in the minority class, respectively. In experiment 5_*, a upper Luhn cut-off value of 808 as well as new users defined stopwords were also used to reduce data dimensionality.

Experiments 2 and 4 show better results for $\mathcal{CN}2$ and $\mathcal{S}$ee5, respectively. Next, it was decided to run some experiments using only the attributes that appear in the rule sets induced by both classifiers in experiment 4. The results obtained are shown in experiment 9. As can be observed a data set with 22214 attributes was reduced, using only attributes that appear in the rule sets induced by both algorithms, to a data set with 19 attributes able to induce classifiers with lower error rate. This small number of attributes represents 0.09% of the original number of attributes. Considering the error of the majority class, the accuracy of this last experiment is very good for this data set and the learning algorithms used.

Aiming to illustrate the performance of a SVM algorithm on this data set using different number of attributes, the attribute-value table from some of the experiments, specifically experiments 1, 5_* and 9, were selected and submitted to the SVMTorch II algorithm using the linear kernel function and default parameters. The results obtained are shown in Table 4.

Table 4. SVMTorch experimental results

Exp	m	M	Att	Ms	10-fold_error %
				tf	0.42 ± 1.3
1	6		6202	tn	0.40 ± 1.3
				tf	0.82 ± 1.7
5_*	30	808	2412	tn	0.40 ± 1.3
				tf	0.00 ± 0.0
9	6	808	19	tn	0.00 ± 0.0

The experimental results show that SVMTorch achieved an excellent performance on this data set. Despite SVMs generally obtain a good performance and eliminate the need for attribute selection, saving a complicated preprocessing step (Joachims, 2002), the excellent results using the data set created by PRETEXT may be explained if it is considered that several irrelevant attributes were discarded. But, it is important to underline that the SVM output is a *black box* classifier. There are situations where it is more important to have a symbolic model (classifier) induced, although less accurate, that allows to interpret, analize and use the knowledge in different ways, such as to organize a smaller index function to retrieve classified documents more efficiently.

6 Conclusion

Considering the unstructured form of textual data, it is notorious the requirement of additional documents preprocessing in the Text Mining process. A primordial question is to determine 'what' and 'how' attributes discriminate the documents considering the high dimensionality of the attribute set.

This work proposes a way to improve the accuracy of learning algorithms through the reduction of the dimensionality of text representation. Our proposal is illustrated on a real case using several learning algorithms and PRETEXT, a computational tool we have implemented. Although not shown in details in the paper due to lack of space, we consider that the facilities that allow the user to interrogate PRETEXT to analize the frequency of words and stems in the whole collection as well as in any document, are important attributes of this tool to further reduce the dimensionality of text representation. The final output of PRETEXT consists of the data file (`.data`) and attributes file (`.names`) in the DISCOVER standard syntax. These two files can be automatically converted to

the attributes and data syntax used by more than sixteen learning systems, such as $\mathcal{S}$See5 and $\mathcal{CN}2$ used in this work, using the computacional environment DISCOVER LEARNING ENVIRONMENT - DLE implemented in the DISCOVER (Batista & Monard, 2003), where PRETEXT is integrated.

Acknowledgments: We thank an anonymous referee for his/her insightful comments. This research is partially supported by Federal Agency for Graduate Education (CAPES) and Universidade Federal de Mato Grosso (UFMT), Brazil.

References

Apté, C., F. Damerau, & S. M. Weiss (1994). Automated learning of decision rules for text categorization. *Information Systems 12*(3), 233–251.

Banerjee, S. & T. Pedersen (2003). The design, implementation and use of the ngram statistics package. In A. Gelbukh (Ed.), *CICLing-2003, LNCS 2588*, pp. 370–381. Springer-Verlag.

Baranauskas, J. A. & M. C. Monard (2000). An unified overview of six supervised symbolic machine learning inducers. Technical Report 103, ICMC-USP. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_103.ps.zip`.

Batista, G. E. A. P. A. & M. C. Monard (2003). Architecture and implementation description of the computational environment DISCOVER LERNING ENVIRONMENT - DLE. Technical Report 187, ICMC-USP. `ftp://ftp.icmc.sc.usp.br/pub/BIBLIOTECA/rel_tec/Rt_187.zip` (in Portuguese).

Batista, G. E. A. P. A. (2003). Data preprocessing in supervised machine learning. PhD Thesis, ICMC-USP (in Portuguese).

Clark, P. & R. Boswell (1989). The $\mathcal{CN}2$ induction algorithm. *Machine Learning 3*(4), 261–283.

Collobert, R. & S. Bengio (2001). SVMTorch: Support vector machines for large-scale regression problems. *Journal of ML Research 1*, 143–160.

Dumais, S., J. Platt, D. Heckerman, & M. Sahami (1998). Inductive learning algorithms and representations for text categorization. In *Proc. of the 7th International Conference on Information and Knowledge Management (CIKM 98)*.

Gelbukh, A. & G. Sidorov (2001). Zipf and heaps laws coefficients depend on language. In *Proc. CICLing2001, Conference on Intelligent Text Processing and Computational Linguistics, LNCS 2004*.

Joachims, T. (2002). *Learning to Classify Text Using Support Vector Machines*. Kluwer Academic Publishers.

Knuth, D. E. (1973). *The Art of Computer Programming*, Volume 3. Addison-Wesley.

Lewis, D. D. (1992, June). An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 37–50.

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development 2*(2), 159–165.

Matsubara, E. T., C. A. Martins, & M. C. Monard (2003). The design and use of the PreTexT tool for text processing. Technical report, ICMC-USP. (to be published).

Montes-y-Gómez, M., A. Gelbukh, A. López-López, & R. Baeza-Yates (2001). Flexible comparison of conceptual graphs. In H. Mayr, J. Lazansky, G. Quirchmayr, & P. Vogel (Eds.), *Proc. DEXA-2001, 12th International Conference and Workshop on Database and Expert Systems Applications. LNCS 2113*, pp. 102–111. Springer-Verlag.

Montes-y-Gómez, M., A. Gelbukh, & A. López-López (2002). Detecting deviations in text collections: An approach using conceptual graphs. In *Proc. MICAI-2002: Mexican International Conference on Artificial Intelligence. LNAI 2313*. Springer-Verlag.

Porter, M. (1980). An algorithm for suffix stripping. *Program 14*(3), 130–137.

Prati, R. C. (2003, april). The integration framework of the DISCOVER system. Master Thesis, ICMC-USP (in Portuguese).

Rulequest-Research (1999). Data Mining Tools $\mathcal{S}$See5 and $\mathcal{C}5.0$. `http://www.rulequest.com/see5-info.html`.

Sebastiani, F. (2002, March). Machine learning in automated text categorisation. *ACM Computing Surveys 34*(1), 1–47.

Van Rijsbergen, C. J. (1979). *Information Retrieval, 2nd edition*. Dept. of Computer Science, Univ. of Glasgow.

Vapnik, V. N. & A. Chervonenkis (1971). On the uniform convergence of relative frequencies of events to their probabilities. Theory of probability and its applications. (16), 262–280.

Wall, L., T. Christiansen, & R. L. Schwartz (1996). *Programming in PERL*. O'Reilly,Inc.

Zipf, G. (1949). *Human Behaviour and the Principle of Least Effort*. Addison-Wesley.