Classification of Macedonian News Articles

Martin Saveski[1], Igor Trajkovski[2], Jovan Pehcevski[3]

[1] Jožef Stefan Institute, Ljubljana, Slovenia
martin.saveski@ijs.si
[2] Ss. Cyril and Methodius University, Skopje, Macedonia
itrajkovski@feit.ukim.edu.mk
[3] European University, Skopje, Macedonia
jovan.pehcevski@eurm.edu.mk

***Abstract*—Due to the increased number of documents in digital form, the need for automatically analyzing and organizing them has rapidly emerged. One of the problems, which due to the wide range of real-world scenarios had raised a lot of interest in the computer science community, is the task of Text Classification. Text Classification has been extensively studied on English document collections; however, to the best of our knowledge no attempts have being made to develop models for classification of Macedonian text documents. In this paper, we present the process of collecting a large text collection of Macedonian news documents (103,637 classified in 6 categories), on which we apply state-of-the-art text classification algorithms. These include: Naïve Bayesian, K-Nearest-Neighbors, Decision Trees, Neural Networks learning algorithms and Ensembles. Our experiments on the Macedonian text collection show that Ensemble of Naïve Bayesian classifiers achieves best results (77.8%) with a slight improvement over the single Naïve Bayesian classifier (75.6%).**

***Keywords*—Text Classification, Macedonian News Collection, Naïve Bayesian, Decision Trees, K Nearest Neighbors, Neural Networks, Ensembles.**

I. Introduction

The rapid growth of the technology has made it easy to store large amounts of data. The increased number of documents available on the Web, news services, corporate intranets, and elsewhere is overwhelming. This data comes in various forms: databases (tabular), images, networks, and other, but most of the data comes at textual form. However, even though the amount of data available to us is constantly increasing, our ability to absorb and process this information remains constant.

Text mining [1] is a well-developed field of computer science that tries to solve the problem of information overload by using techniques from data mining, machine learning, natural language processing (NLP), information retrieval (IR), and knowledge management. In a manner analogous to data mining, text mining seeks to extract useful information from data sources through identification and exploration of interesting patterns. Unlike in data mining, in the case of text mining the data sources are document collections, and interesting patterns are found not among formalized database records, but rather in the unstructured textual data.

In text mining various supervised, semi-supervised, and unsupervised learning principles are employed to perform higher-level tasks such as text classification, text clustering, topic ontology construction, text corpora visualization [2], user profiling [3], etc. One of the most commonly studied tasks is the problem of classification or categorization of textual documents. Given a set of categories (subjects, topics) and a collection of textual documents, Text Classification is the process of finding the correct topic (or topics) for each document in the collection. Text Classification has been extensively used in various real-life applications including: indexing of scientific articles, patent categorization, spam filtering, Web page classification under hierarchical catalogues, automatic generation of metadata, detection of text genre, and many others [4].

While the task of Text Classification of document collections in English has been well covered in the Text Mining literature, we do not know of any attempt made to study the problem of classification of textual documents in Macedonian. In this paper, we present the process of collecting a large textual corpus of Macedonian news articles from the Web (103,637 classified in 6 categories), on which we investigate various text classification algorithms that have been used to develop a model for classification of Macedonian text documents. We conducted experiments using the following algorithms: Naïve Bayesian, K-Nearest-Neighbors, Decision Trees, and Neural Networks learning algorithms [1, 10]. We discuss the process of Feature Selection and building Ensembles of classifiers in order to improve the performance accuracy of the resulting models. Furthermore, we also present details about the implementation of the algorithms and indicate problems and challenges we faced when applying them to large document collections.

The rest of the paper is organized as follows. In Section 2, we present the process of building the textual document collection. The different text classification algorithms are presented in Section 3. Finally, in Sections 4 and 5, we present the results of our experiments and provide conclusions.

II. Collecting the Data Set

The process of collecting the data set comprises the following steps: crawling, boiler plate removal and pre-processing of the data.

CRAWLING: For the purpose of the experiments for this study we have decided to use the news data available from the A1 TV[1] news archive. We considered this source as most appropriate since it has the largest archive and is most accessible i.e. easiest to crawl. To collect the data we have developed a simple multithreaded crawler application in Java. To ensure maximum performance, the application was configured to open 25 threads on every 5 seconds, where each thread sends a separate HTTP request, receives the corresponding html document, and extracts the actual text content from the page.

BOILER PLATE REMOVAL: After each page is crawled we have used a set of regular expressions to automatically extract the title, and the category from the raw the HTML page. Although we are aware of many advanced and general Machine Learning algorithms for Boiler Plate Removal [5] (extracting text from and HTML page), because we are dealing with only one news source we have opted for much simpler technique. Namely, to extract the actual text content from the HTML page we first represented the character array (string) of the HTML document as an integer array, where every Cyrillic letter is given weight 1 and every other letter is given weight -1. From this array, we calculate the maximum sub sum, where the start and end position of the maximum sub sum are the start and end position of the news text. This simple algorithm extracts the largest chunk of connected Cyrillic letters and assumes that this is the actual content of the news page and that the rest is an HTML code making the page design template.

PRE-PROCESSING: After the news text is extracted, the HTML tags are removed and the HTML entities are converted to their equivalents. To be able to access the news texts more efficiently, we have stored the texts in MySQL database with one simple table with title, text, and category as attributes. The crawling of the whole archive took about ten hours. Finally, we have preprocessing the texts by: cleaning the punctuation of every text, tokenization, stop word removal, and converting to lower case. We also assigned a unique word id to every word appearing in the collection and we converted each text to a hash table where the word ids are keys and their TF-IDF scores are values. This last step allowed us to work only with integers and to easily manipulate with the texts. Table 1 shows the whole dataset containing 100,637 articles i.e. 112,499,196 tokens classified in 6 categories, published between January 2001 and December 2009.

Category	Articles	Tokens
Balkan	15,264	1,859,956
Economy	13,053	1,560,579
Macedonia	34,323	5,385,368
Sci/Tech	4,920	127,775
World	18,845	2,122,560
Sport	17,232	1,442,958
TOTAL	**103,637**	**12,499,196**

Table 1. A1 Corpus, size and categories

III. Text Classification Algorithms

We now discuss the theoretical background of the text classification algorithms employed during the experiments. In subsections *A* and *B*, we present the Naïve Bayesian and K-Nearest-Neighbors classification algorithms, respectively. As a most suitable method to deal with the large dimensionality encountered in text classification in subsection *C* we discuss the various features selection techniques. In subsections *D* and *E*, we present the Decision Trees and Neural Networks learning algorithms. Finally, in subsection *F* we discuss the use of Ensembles to combine several classification models.

A. *Naïve Bayesian*

This learning algorithm is very efficient, easy to implement, and has already been shown that it produces successful results in classification of English text articles [6]. In the training phase, for each category the frequency of each word is calculated, along with the total number of words in each category, the number of texts in each category, and the total number of documents. According to this information the new test texts can be classified. As the equation below shows, the number of texts in each category ($|D_i|$) over the total number of texts ($|D|$) give the prior probability of the categories and product of the frequency of each word in the text (n_{ij}) over the total number of words in this category (n_i) give the posterior probability.

$$P(C) = \frac{|D_i|}{|D|}, P(w_j|c) = \frac{n_{ij}+1}{n_i} \quad (1)$$

To overcome the zero probabilities for the words that are not seen in the training data, every new word is given frequency of 1, even it has not occurred in the training set. Also, to overcome the problem of underflow log likelihood was used. The maximum of the logarithms of the probabilities is also the maximum of the probabilities for each category. As it can be seen from the formula the algorithm does not consider the order of the words in the text and assumes independence between the individual words.

[1] http://www.a1.com.mk/

Once the probability estimations are calculated from the training set, the unlabeled examples are classified in the category with the largest probability:

$$argmax_{C_i \in C} P(C_i) \prod_{i=1}^{n} P(a_i \mid c_i) \quad (2)$$

where *C* is the set of categories and *a* is the set of words in the text.

B. KNN - K Nearest Neighbors

This classifier belongs to the family of Instance Based classifiers. The classification consists of finding the K most similar training samples with the test sample (nearest neighbors) and returning the most common class where they belong. This approach of nonparametric learning allows the complexity of the hypothesis to grow with the complexity of the data. The KNN classifier essentially has no training phase. However, if the data samples are not structured to allow the neighbors to be found more efficiently, to classify one document the similarity with every other document must be measured. This requires a lot of time and processing. For this reason during the training phase we built an inverted-index. Inverted-indexing is a well-known technique commonly used in Information Retrieval and it consist of a building a hash table where each word in dataset dictionary is mapped to a set of documents in which it appears. Thus, when classifying the new samples a set of most probable neighbors is formed. For each word in the document only the most characteristic documents (according to the inverted-index) in the training samples are added to the set. This significantly decreases the number of comparisons per document. To further decrease the time needed for classification from every text in the train corpus only the most important 10 words were retained (sorted by TF-IDF) and the inverted index was build to contain only at most 30 documents for each word. To measure the similarity between two documents the Cosine Similarity measure was used. The Cosine Similarity is a classical approach for comparing text documents where the similarity between two documents is defined as the cosine of the angle between the two document vectors. However, despite the various enhancements, the classifier takes long time to classify the large set of test samples. Also, requires a lot of memory since every text and the whole inverted index must be kept in memory.

C. Feature Selection

The first two classifying algorithms can tolerate the large number of feature imposed by the task of text classification. Since a stemmer for the Macedonian language is not publicly available, the feature space is very large. On all 103,637 documents there are 271,538 features (unique words). It's obvious that there must be selection of the features so that decision trees and neural networks can be built. Yang and Pedersen in [7] explain and compare several approaches to feature selection for text classification including the following: document frequency, information gain, mutual information, chi test, and term strength. Their analysis shows that information gain and chi test are most effective in aggressive term removal, up to 98% of the features. However, they have also concluded that the document frequency measure achieves the same result when reducing up to 90% of the features. The chi test and information gain measures have O(Vm) time complexity (where V is the words in the vocabulary and m is the number of categories), while the document frequency has O(V) time complexity. In addition, the document frequency measure is very simple to implement and can easily scale to the large corpora. For this reason, we have decided to use the document frequency measure for feature selection.

D. Decision Trees

To apply this classification algorithm the features of each text were represented as Booleans. A feature has value true if a given word appears in the text and false otherwise. This simplifies the algorithm, but comes at a cost of decreased performance. Another approach is to have three (or more) values for the features, for example: didn't appear, low frequency, and high frequency. However, as discussed in the previous section, even with the features space reduced building a decision tree still requires a large amount of time. For example, building a decision tree on a set of 500 features requires about one and a half hour on a standard desktop machine[2]. For this reason, instead of building one single decision tree, we decided to build an ensemble of decision trees (we call it decision forest) which are trained on different sets of features, but on the same set of training examples. Thus, each decision tree has limited depth and is built faster. Each decision forest contains 5 decision trees with 1000 features and 30 decision trees with 500 features, thus classifying the documents by 20,000 unique features. To classify new unseen documents, all decision trees vote for a category. But, not all decision trees are equally competent to classify the document. Thus, based on how many of the features in the document are known by the tree a weight for its vote is given and the category of the document is determined by weighted majority vote.

To determine the order in which the features are tested in the decision trees, the information gain heuristic was used. In addition to allow the decision trees to generalize well, if there are too few examples in a node of the tree the most common class was returned.

E. Neural Networks

Text document classification has high dimensional data characteristics because of the large size of natural language vocabulary. Documents in one class usually can be linearly separated from other classes due to the high dimensionality [8]. Accordingly, linear neural networks can achieve the same accuracy as non-linear neural networks with hidden layers [9]. In addition, the linear neural networks require less memory and allow more features to be used for classification. For these

[2] 4GB RAM, 2.4 Dual Core Processor

reasons, we have built a neural network with 16,000 input nodes and 8 output nodes. The input nodes correspond to document features i.e. how many times the word appears in the document and the output nodes correspond to estimated categories probabilities. During the learning process, the back propagation algorithm was used to minimize training error after each document. Also, instead of a threshold function we have used the sigmoid (logistic function). During the experiments we found that the learning rate of *0.05* is most effective for the learning process. We also considered the possibility of adding momentum to the learning algorithm to speed up the learning process, but the time frame did not allow this to be implemented.

Due to the massive calculations required by the algorithm the initial implementation developed in Python[3] showed to be too slow, so we decided to re-implement the algorithm in C++. For comparison, the C++ implementation reduced the time needed for one epoch by factor of 200, taking only 10 minutes to execute an epoch. To achieve reasonable error (below 1000 on 85 000 documents) about 50 epochs were needed.

F. Ensemble

The idea of ensembles is to select a collection of hypothesis and to combine their predictions. This can be seen is a generic way of enlarging the hypothesis space. For this task the Bagging method was used for the development of the ensembles. We experimented with two methods for selecting the training samples for the hypothesis. The first one divides the training set on N portions (given N hypothesis are to be produced), then each classifier is trained on all samples except for one portion, where this portion is different for each classifier. The second approach consists of selecting random documents for the training samples to produce different training samples for each classifier so that each produced training set contains the same number of sample as the original one. In our preliminary experiments both methods showed similar results, and so only the results from the first approach are reported. The classification of new samples is done using majority voting among all hypotheses in the ensemble. The methods implemented for building and testing ensembles are general and can be used for any of the other classifiers just by specifying the appropriate training and testing methods. To compare the performance of the ensembles of classifiers versus single one the Naïve Bayesian classifier was used.

IV. Experimental Results

To measure the performance of each classification algorithm we have used three measures: precision, recall, and F-measure. Precision (P) is defined as the number of true classifications in the class over the number of documents classified in the class, recall (R) is defined as the number of true classifications in the class over the total number of document in the class. F-measure is a combination of precision and recall giving a single global measure of performance, which is calculated as follows:

$$F = 2 * \frac{P * R}{P + R}$$

The chart shown in Fig. 1 below graphically depicts the performance of the classifiers with 10-fold cross-validation. The best performance of the single classifiers is achieved by the Naïve Bayesian classifier, while the ensemble of 100 such classifiers further improves the performance by 3.2% (F-measure). Specifically, an ensemble of 100 hypotheses showed a performance of 77.8%, which is however not significant improvement over the performance of a single Naïve Bayesian classifier. The decision trees seem to be prone to over fitting of the training data and not being able to generalize well showing very low performance of 38.6%. It is interesting to note the high precision rate (80.6%) of the Neural Networks; however the low recall significantly decreases the overall performance. Considering both the performance and the efficiency the Naïve Bayesian classifier is dominant and shows to be the best choice for the task. Although the ensemble of NB classifiers shows slight improvement, we believe that the additional processing introduced makes this solution less appropriate for use in real-world scenarios.

V. Conclusions and Future Work

In this paper we discussed one of the most important Text Mining problems, the task of Text Classification, with relevance to Macedonian textual documents collections. We presented the process of crawling a large textual document collection of Macedonian news articles and the steps of its pre-processing relevant to the text classification task. We also discussed several state-of-the-art classification algorithms and demonstrated how they can be used in the context of Text Classification. Finally, we presented the results of the experiments we conducted to measure the performance of each of the approaches for document classification. We found that the Ensemble of Naïve Bayesian classifiers achieved the best performance with slight improvement over the single Naïve Bayesian classifier.

In future work, we are interested in analyzing how more advance natural language processing (NLP) techniques, such as stemming, lemmatization or part-of-speech tagging can be used to pre-process the document to reduce the document feature space, and how this reduction will influence the performance of the Text Classification algorithms.

[3] http://www.python.org/

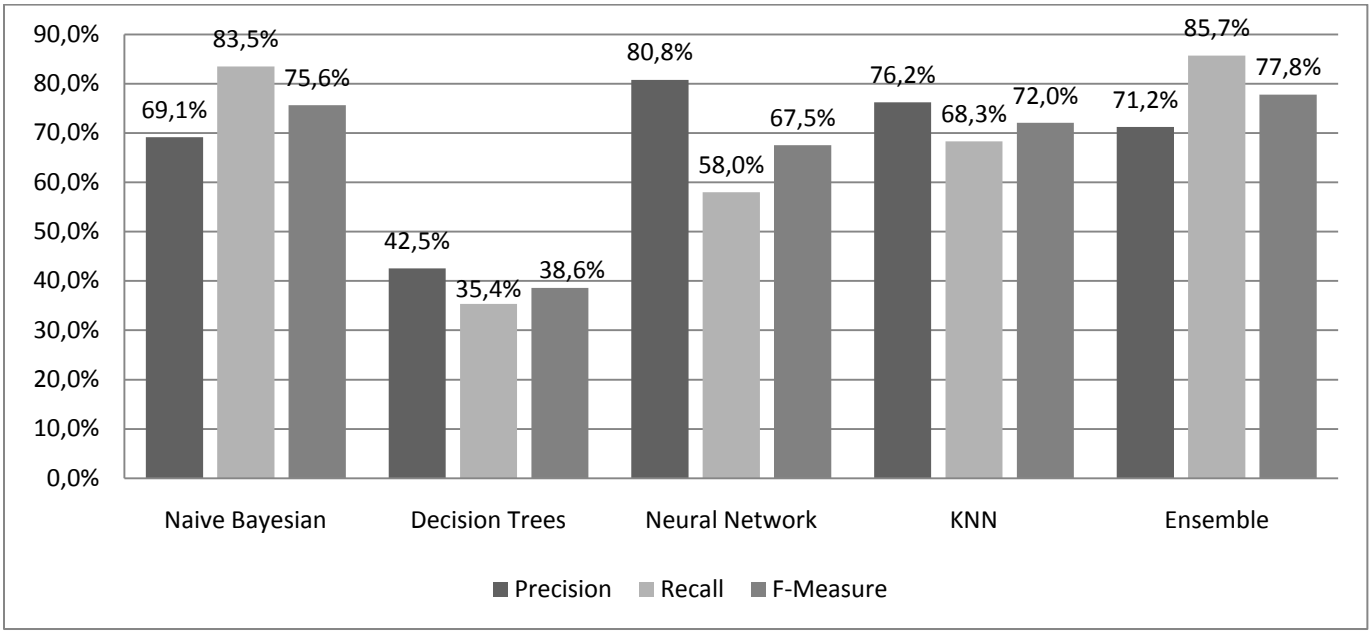

Fig 1. Comparison between the performance of the different text classification algorithms

REFERENCES

[1] R. Feldman, J. Sanger. *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2006.

[2] B. Fortuna, M. Grobelnik, D. Mladenic. *OntoGen: Semi-Automatic Ontology Editor*. HCI International '07, Beijing, July 2007.

[3] H. R. Kim, P. K. Chan. *Learning Implicit User Interest Hierarchy for Context in Personalization*. Journal of Applied Intelligence, vol. 28, issue 2, 2008.

[4] F. Sebastiani. *Machine Learning in Automated Text Categorization*. ACM Computing Surveys, vol. 34, issue 1, pages 1–47, 2002.

[5] C. Kohlschütter, P. Fankhauser, W. Nejdl. *Boilerplate Detection using Shallow Text Features*. Proceedings of The Third ACM International Conference on Web Search and Data Mining, WSDM 2010. New York, 2010.

[6] I. Rish. *An empirical study of the naive Bayes classifier*. IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence. 2001.

[7] Y.Y. Yang, J.O.P. Pedersen. *A Comparative Study on Feature Selection in Text Categorization*. Proceedings of ICML-97: p. 412-420, 1997.

[8] C.L. Ciya, S.A. Alpha, P.D. Dixon. *Feature Preparation in Text Categorization*. Oracle Corporation. 2007.

[9] T.J. Joachims. *Text categorization with support vector machines: learning with many relevant features.* Proceedings of ECML-98, 10th European Conference on Machine Learning, 1998.

[10] S. Russell, P. Norvig. *Artificial Intelligence: A Modern Approach*. 2002.