Distributed Classification of Textual Documents on the Grid

Ivan Janciak[1], Martin Sarnovsky[2], A Min Tjoa[3], and Peter Brezany[1]

[1] Institute of Scientific Computing, University of Vienna, Nordbergstrasse 15/C/3 A-1090 Vienna, Austria
janciak@par.univie.ac.at, brezany@par.univie.ac.at

[2] Department of Cybernetics and Artificial Intelligence, Technical University of Kosice, Letna 9, Kosice, Slovakia
martin.sarnovsky@tuke.sk

[3] Institute of Software Technology and Interactive Systems, Vienna University of Technology, Favoritenstrasse 9-11/E188, A-1040 Vienna, Austria
tjoa@ifs.tuwien.ac.at

Abstract. Efficient access to information and integration of information from various sources and leveraging this information to knowledge are currently major challenges in life science research. However, a large fraction of this information is only available from scientific articles that are stored in huge document databases in free text format or from the Web, where it is available in semi-structured format.

Text mining provides some methods (e.g., classification, clustering, etc.) able to automatically extract relevant knowledge patterns contained in the free text data. The inclusion of the Grid text-mining services into a Grid-based knowledge discovery system can significantly support problem solving processes based on such a system.

Motivation for the research effort presented in this paper is to use the Grid computational, storage, and data access capabilities for text mining tasks and text classification in particular. Text classification mining methods are time-consuming and utilizing the Grid infrastructure can bring significant benefits. Implementation of text mining techniques in distributed environment allows us to access different geographically distributed data collections and perform text mining tasks in parallel/distributed fashion.

Keywords: Text Mining, Multi Label Text Categorization, Distributed Text Mining, Grid Computing, JBOWL, GridMiner.

1 Introduction

The process of data mining is one of the most important topics in scientific and business problems. There is a huge amount of data that can help to solve many of these problems. However, data are often geographically distributed in various locations. While text is still premier source of information on the web, the role of text mining is increasing.

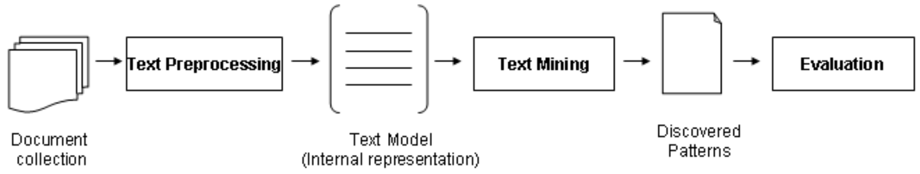

Fig. 1. Process of knowledge discovery in textual documents

Nowadays, the information overload means a big problem, so text mining algorithms working on very large document collections take very long times on conventional computers to get results. One approach to face this problem is distributed computing - distributed text mining algorithms can offer an effective way to mine extremely large document collections.

Motivation of this work is to use the Grid computational capabilities to solve text mining tasks. Some of the methods are time-consuming and use of the Grid infrastructure can bring significant benefits. Implementation of text mining techniques in distributed environment allows us to perform text mining tasks, such as text classification, in parallel/distributed fashion.

Knowledge discovery in texts is a variation on a field called knowledge discovery in databases, that tries to find interesting patterns in data. It is a process of semiautomatic non-trivial extraction of previously unknown, potentially useful and non-explicit information from large textual document collection, as depicted on Figure 1. A key element of text mining is to link extracted information together to form new facts or new hypotheses to be explored further by more conventional means of experimentation. While regular data mining extracts the patterns from structured databases of facts, text mining deals with problem of natural language processing. The biggest difference between data mining and text mining is in the preprocessing phase. Preprocessing of text documents is completely different, in general, it is necessary to find a suitable way to transform the text into an appropriate internal representation, which the mining algorithms can work on. One of the most common internal representations of document collection is the *Vector Space Model* [6]. Text mining phase is the core process of knowledge discovery in text documents. There are several types of text mining tasks as follows:

– Text categorization : assigning the documents into the pre-defined categories
– Text clustering : descriptive activity, which groups similar documents together
– Information retrieval : retrieving the documents relevant to the user's query
– Information extraction : question answering.

Nowadays, text mining plays important role in the area of processing biomedical databases that contain huge volume of textual documents. For example, one of the current questions in genomics is to inspect which proteins interact with others. There has been notable success in looking at which words co-occur in articles that discuss the proteins in order to predict such interactions. The key is not to search for direct occurrence of pairs in document, but to find articles that mention individual protein names and keep track of which other words occur in those articles, and finally look for other articles containing the same sets of words. This method can yield surprisingly good results, even though the meaning of the texts are not being discerned by the programs.

The structure of the rest of the paper is organized as follows. Section 2 discusses the classification of documents using multi-label classification. Section 3 describes the design of sequential and distributed versions of the text mining service implemented using the knowledge discovery framework - GridMiner. Experimental performance results are discussed in Section 4. Related work is presented in Section 5 and we briefly conclude in Section 6.

2 Text Classification Based on a Multi-label Algorithm

Text Classification is the problem of assigning a text document into one or more topic categories or classes based on document's content. Traditional approaches to classification usually consider only the unilabel classification problem. It means that each document in collection has associated one unique class label.

This approach is typical for data mining classification tasks, but in a number of real-world text mining applications, we face the problem of assigning the document into more than one single category. One sample can be labeled with a set of classes, so techniques for the multi-label problem have to be explored. Especially in text mining tasks, it is likely that data belongs to multiple classes, for example in context of medical diagnosis, a disease may belong to multiple categories, genes may have multiple functions, etc.

In general, there are many ways to solve this problem. One approach is to use a multinomial classifier such as the Naive Bayes probabilistic classifier [4], that is able to handle multi-class data. But most of common used classifiers (including decision trees) cannot handle multi-class data, so some modifications are needed. Most frequently used approach to deal with multi-label classification problem is to treat each category as a separate binary classification problem, which involves learning a number of different binary classifiers and use an output of these binary classifiers to determine the labels of a new example. In other words, each such problem answers the question, whether a sample should be assigned to a particular class or not.

In the work reported in this paper, we used the decision trees algorithm based on the Quinlan's C4.5 [7]. A decision tree classifier is a tree with internal nodes labeled by attributes (words), branches are labeled by weight of the attribute in a document, and leafs represent the categories [1]. Decision tree classifies a sample by recursively testing of the weights in the internal nodes until a leaf is reached.

While this algorithm isn't suitable to perform multi-label classification itself, we use the approach of constructing different binary trees for each category.

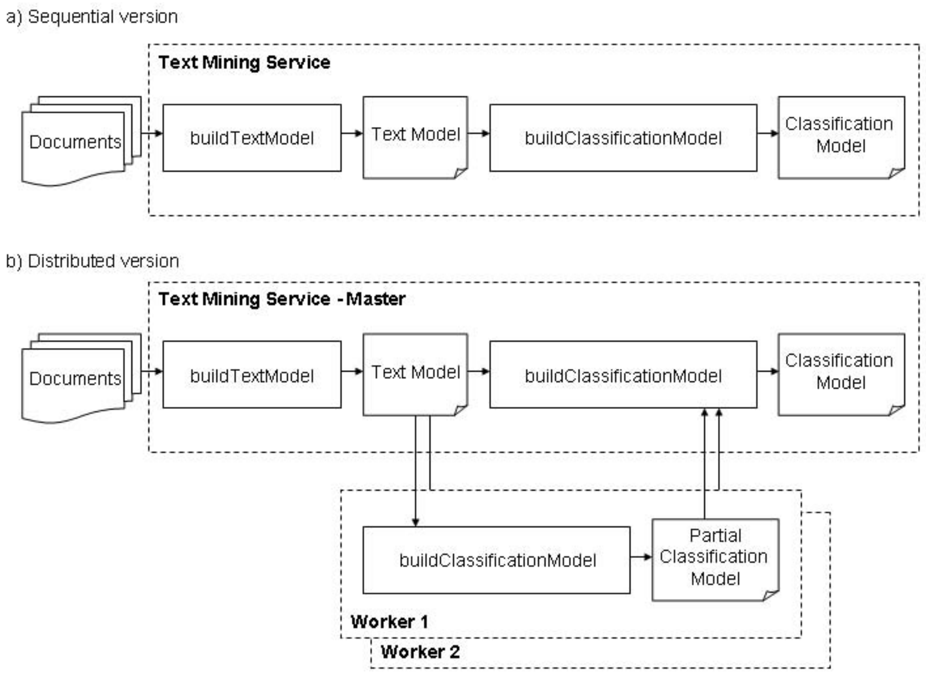

Fig. 2. Design of sequential and distributed text mining services

The process of building many binary trees can be very time consuming when running sequentially, especially on huge document collections. Due to the fact that these binary classifiers are independent on each other, it is natural to find a suitable way how to parallelize the whole process. Growing of these binary trees is ideal for parallel execution on a set of distributed computing devices. Such a distribution might be desirable for extremely large textual document collections or large number of categories, which e.g. can be associated with a large number of binary classifiers.

3 Architecture and Implementation of the Classification Service

Building distributed text mining services is an inherently difficult and complex task. To reduce the complexity, the first goal was to design a sequential version of Text Classification Service based on the multi-label algorithm implemented in the JBOWL library, which is discussed below.

3.1 JBOWL

JBOWL - (Java Bag-of-Words Library) [2] is an original software system developed in Java to support information retrieval and text mining. The system is being developed as open source with the intention to provide an easy extensible, modular framework for pre-processing, indexing and further exploration of large text collections, as well as for creation and evaluation of supervised and unsupervised text-mining models. JBOWL supports the document preprocessing, building the text mining model and evaluation of the model. It provides a set of classes and interfaces that enable integration of various classifiers. JBOWL distinguishes between classification algorithms (SVM, linear perceptron) and classification models (rule based classifiers, classification trees, etc.).

3.2 GridMiner

GridMiner [3] is a framework for implementing data mining services in the Grid environment. It provides three layered architecture utilizing a set of services and web applications to support all phases of data mining process. The system provides a graphical user interface that hides the complexity of the Grid, but still offers the possibility to interfere with the data mining process, control the tasks and visualize the results. GridMiner is being developed on top of the Globus Toolkit[1].

3.3 Implementation

The interface of the sequential and distributed versions of the service defines two main methods needed to build final model: *BuildTextModel* and *BuildClassificationModel*. While the first one is implemented as a pure sequential method, the second one can build the final model distributing the partial binary classifiers. This behavior of the service depends on its configuration. A simplified architecture of both versions is depicted in Figure 2. Moreover, other methods were implemented to provide term reduction and model evaluation, but these methods were not used during the performance evaluation experiments discussed in Section 4.

1. **BuildTextModel** - This method creates the *Text Model* from the documents in the collection. The model contains a document-term matrix created using TF-IDF weighting [8], which interprets local and global aspects of the terms in collection. The input of the method is a parameter specifying the text model properties and the location of the input collection.
2. **BuildClassificationModel** - The *Classification Model*, as the result of the decision tree classifier, is a set of decision trees or decision rules for each category. This service method creates such a model from the document-term matrix created in the previous method. The sequential version builds the model for all categories and stores it in one file. The process of building the model iterates over a list of categories and for each of them creates a binary decision tree. The distributed version performs the same, but it distributes the work of building individual trees onto other services, so called workers, where partial models containing only trees of dedicated categories

[1] http://www.globus.org

are created. These partial models are collected and merged into the final classification model by the master node and stored in the binary file, which can be passed to a visualization service.

4 Experiments

In this section, we present experiments performed on the local area network of the Institute of Scientific Computing in Vienna. As the experimental test bed, we used five workstations Sun Blade 1500, 1062MHz Sparc CPU, 1.5 GB RAM connected by a 100MBit network.

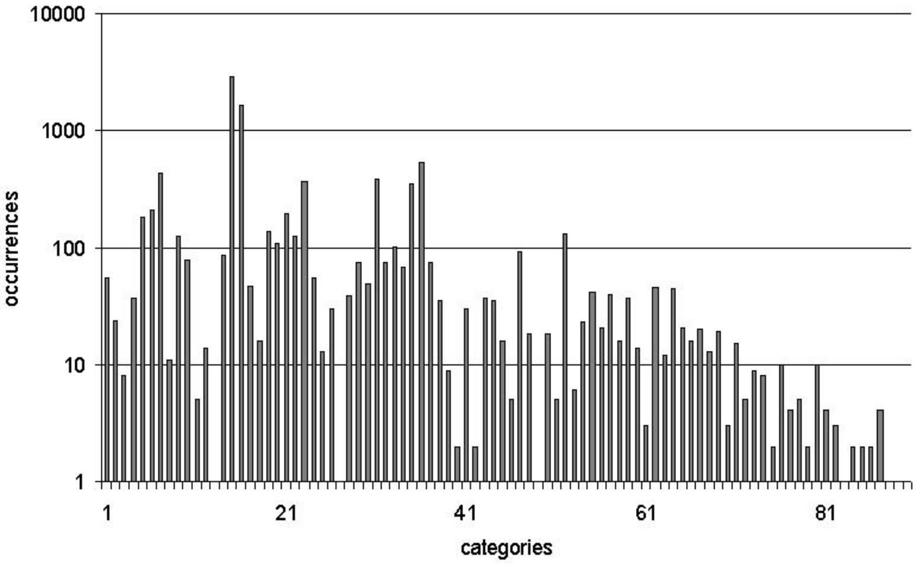

Fig. 3. Logarithmic distribution of categories frequency in the Reuters-21578 dataset

4.1 The Training Dataset

The Reuters-21578 [5] document collection was used as the training data in our experiments. It is de facto a standard dataset for text classification. Its modification, ModApte split [1], consisting out of 9603 training documents in 90 categories and formatted in XML, was used in all tests. Figure 3 depicts the logarithmic distribution of category frequencies in the Reuters-21578 collection. It shows that there are only two categories that occur more than 1000 times in the collection, while there is a lot of categories with frequency of occurrence less than 10. The time needed to build binary classifiers for categories with highest frequency of occurrence is significantly longer than the building time for the rest of the categories. This is a key factor for tasks distribution and optimization. In our case it is a decision how to assign partial categories and associated binary decision trees construction to the worker nodes.

4.2 Performance Results

The main goal of the experiments was to prove, that the distribution of processes mentioned above, can reduce the time needed to construct the classification model. We started the experiments using the sequential version of the service, in order to compare the sequential version with the distributed one. The time to build the final classification model on a single machine using the ModApte dataset was measured three times and its mean value was 32,5 minutes. Then we performed the first series of the distributed service tests without using any optimization of distribution of categories to the worker nodes. According to the number of worker nodes, the master node assigned the equal number of categories to each worker node. The results, see Figure 4, show us the speedup of building the classification model using multiple nodes. The detailed examination of the results and the document collection proved that the time to build a complete classification model is significantly influenced by the working time of the first node. Examination of the dataset and workload of particular workers showed us that the first node always received a set of categories with the highest frequency of occurrences in the collection. It means that other worker nodes always finished the building of their partial models in a shorter time than the first one. It is caused by non-linear distribution of category occurrences as discussed in Section 4.1. The most frequent category (category number 14) occurs in 2780 documents and it was always assigned to the first worker node. That was the reason, why the first worker node spent much longer time to build-up the partial model.

After the first series of tests, we implemented the optimization of distribution of the categories to the worker nodes according to the frequency of category occurrences in the documents. Categories were sorted by this frequency and distributed to the worker nodes according to their frequency of occurrence, what means that each node was assigned with equal number of categories, but with a similar frequency of their occurrences.

We run the same set of the experiments as in the first series and the results showed us more significant speedup using less worker nodes, see optimized bars in Figure 4. The best performance results were achieved using optimized distribution on 5 worker nodes (5.425 minutes), which was comparing to single machine computing time (32.5 minutes) almost 6 times faster. The minimal time to complete classification model is limited by the time of processing of the most frequent category - if this is assigned to a single worker node.

5 Related Work

In this section, we describe projects utilizing the Grid to perform advanced knowledge discovery in textual documents. DiscoveryNet[2] provides a service oriented computing model for knowledge discovery, allowing the user to connect to an use data analysis software as well as document collection that are made

[2] http://www.discovery-on-the.net

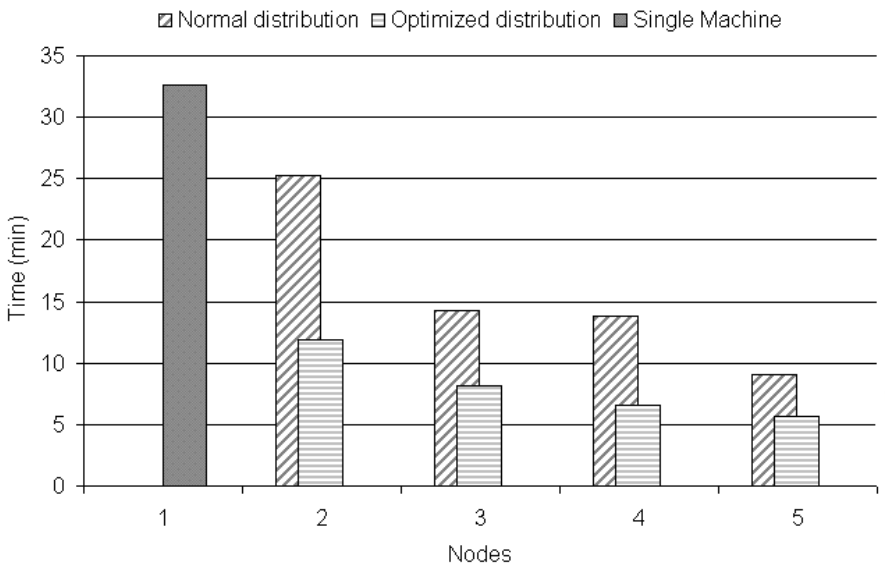

Fig. 4. Performance results of the normal and optimized distribution of nodes workloads

available online by third parties. The aim of this project is to develop a unified real-time e-Science text-mining infrastructure that leverages the technologies and methods developed by the DiscoveryNet and myGrid[3] projects. Both projects have already developed complimentary methods that enable the analysis and mining of information extracted from biomedical text data sources using Grid infrastructures, with myGrid developing methods based on linguistic analysis and DiscoveryNet developing methods based on data mining and statistical analysis. National Centre for Text Mining[4] is also involved in research activities covering the Grid based text mining. Primary goal of this project is also focused to develop an infrastructure for text mining, a framework comprised of high-performance database systems, text and data mining tools, and parallel computing.

6 Conclusions and Future Work

In this paper we presented a comparative study of sequential and distributed versions of classifiers based on decision trees. We proposed an idea how to distribute the process of building a multi-label classification model in the Grid environment by splitting the set of particular binary classifiers, needed to construct the final models into the workpackages, that are computed in distributed fashion. The results proved that the distributed version can bring significant benefits and

[3] http://www.mygrid.org.uk
[4] http://www.nactem.ac.uk

helps to reduce computing time needed to build the classification model. We also implemented an optimized distribution of particular binary classifiers onto the worker nodes, which had inconsiderable impact on the final time reduction comparing to the non-optimized approach. On a different real-world datasets, the speedup of the distributed version may be more significant. In our future research effort, we plan to explore and extend this approach of distribution of text classification services to other text mining tasks.

References

1. C. Apte, F. Damerau, and S. M. Weiss. Towards language independent automated learning of text categorisation models. In *Research and Development in Information Retrieval*, pages 23–30, 1994.
2. P. Bednar, P. Butka, and J. Paralic. Java library for support of text mining and retrieval. In *Proceedings of Znalosti 2005, Stara Lesna*, pages 162–169, 2005.
3. P. Brezany, I. Janciak, A. Woehrer, and A Min Tjoa. Gridminer: A framework for knowledge discovery on the grid - from a vision to design and implementation. In *Cracow Grid Workshop*, Cracow, December 2004.
4. Pedro Domingos and Michael J. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.
5. D. D. Lewis. Reuters-21578 text categorization test collection distribution 1.0. http://www.research.att.com/ lewis, 1999.
6. H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Developement*, 4:309–317, 1957.
7. J. R. Quinlan. Learning first-order definitions of functions. *Journal of Artificial Intelligence Research*, 5:139–161, 1996.
8. G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523, 1988.