PRETO: A High-performance Text Mining Tool for Preprocessing Turkish Texts

Volkan Tunalı, Turgay Tugay Bilgin

Abstract: *Text documents are usually unstructured and written in natural language. To apply conventional data mining techniques on text documents, a preprocessing operation is indispensable. In this paper, we introduce PRETO, a cross-platform, powerful and scalable preprocessing tool developed specifically for preprocessing Turkish texts, with a wide range of preprocessing options like stemming, stopword filtering, statistical term filtering, and n-gram generation. We demonstrate the performance and scalability of PRETO with some experiments on large document collections.*

Key words: *Data Mining, Text Mining, Text Preprocessing, Natural Language Processing.*

1. INTRODUCTION

Advances in information and communication technologies has enabled organizations and individuals to produce and store large amount of information in the form of text documents like news articles, research papers, e-mail messages, letters, reports, surveys, presentations, and web pages [5]. In order to effectively handle this "information explosion" problem and utilize huge amount of text databases, efficient and scalable tools and techniques are necessary. Text mining is the process of extracting previously unknown, potentially useful and valuable information from large amount of textual data [4, 9].

Since text documents are generally unstructured and written in natural language, the very first step in text mining is preprocessing which transforms unstructured text data into structured format. After preprocessing, conventional data mining techniques can be applied on the structured representation of textual data. There are several methods used in text mining for preprocessing text documents. Among these, tokenization, stemming, stopword filtering, and term weighting are most commonly applied [6].

Due to high acceptance and use of information and communication technologies by Turkish people all around the world, textual data in Turkish is also increasing rapidly. Turkish is an agglutinative language where words are constructed using inflectional and derivational suffixes linked to a root. In addition, the Turkish alphabet contains seven letters specific to Turkish language (ç, ğ, ı, İ, ö, ş, ü) and does not contain the letters q, w, and x [2]. Those properties require special treatment while preprocessing texts in Turkish.

In this paper, we introduce PRETO, a standalone, platform-independent preprocessing tool developed specifically for preprocessing Turkish texts. PRETO offers a wide range of preprocessing options like stemming, stopword filtering, statistical term filtering, and n-gram generation. PRETO is a powerful preprocessing tool designed for high speed and scalability with modest main memory consumption in order to preprocess very large document collections in a reasonable amount of time on a single processor.

This paper is organized as follows. Background information on preprocessing is presented in Section 2. In Section 3, we give the details of the design and implementation of PRETO along with some performance and scalability measurements. Finally, Section 4 provides some concluding remarks and describes our future plans on enhancing its capabilities.

CompSysTech'12, June 22-23, 2012, Ruse, Bulgaria.

2. TEXT PREPROCESSING TECHNIQUES

In order to mine text collections, text documents must be preprocessed and stored in more appropriate data structures for further processing. In most text mining applications, essential data model is based on the idea that a text document can be represented by the set of words contained in it. This model is called the Vector Space Model where each document is represented as a vector and each word in a document is represented by a numerical importance or weight value within the vector [6, 8]. After preprocessing is completed, a document-term matrix is obtained, in which rows correspond to document vectors and columns correspond to words (or terms in general). In this section, we briefly present important text preprocessing techniques.

First of all, a text document is split into a sequence of words by ignoring all punctuation marks, white-spaces, tabs, and other non-text characters. This process is called tokenization, and the words in raw form are called tokens. This tokenization step makes the text clean and ready for further processing. The set of different words obtained from all documents of a collection is called the dictionary of a document collection [6].

In order to decrease the size of the dictionary and thus the dimensionality of the document-term matrix, the set of words describing the documents can be reduced by applying stemming and filtering on the words.

Once a sequence of tokens has been obtained from a text document, the next step is to convert each token to a basic form by a process usually referred to as stemming or lemmatization. Stemming, for example, removes the plural "s" from nouns, the "ing" from verbs, or other affixes. Therefore, every word is represented by its stem and number of distinct words in the dictionary is reduced [10]. A well-known and widely used rule based stemming algorithm for English is the Porter's algorithm [7]. For stemming Turkish words, Zemberek can be used. Zemberek is an open source, platform-independent, general purpose Natural Language Processing library and toolset designed for Turkic languages, especially Turkish. Zemberek is officially used as spell checker in Open Office Turkish version and Turkish national Linux Distribution Pardus [1].

Some words like articles, conjunctions, and prepositions convey no valuable content information for text mining operations. Therefore, it is a useful approach to remove these words from the dictionary. This is a standard filtering method called stopword filtering that uses a list of predefined words like "the", "more" in English, and "böyle" (*such*), "daha" (*more*) in Turkish. Moreover, words that appear too often in the whole collection can be thought that they contain no valuable information to distinguish between documents, and also words that occur very rarely can be regarded that they have no statistical benefit. Thus, such words can be removed from the dictionary as a filtering approach [6].

In text mining applications, single words are not the only candidates for terms. Instead, more than one consecutive words can be used as terms because sequence of words may contain more valuable information than single words. A word n-gram is a contiguous sequence of n words from a given sequence of text. An n-gram of size 1 is called a "unigram"; size 2 is a "bigram"; size 3 is a "trigram".

Term weighting is an important part of text preprocessing. In Vector Space Model, simplest method of representing term occurrences is the binary weighting where terms are represented with binary values 0 or 1 depending on the occurrence of a term in document. This method, however, does not consider the importance of terms within document and throughout the collection. Therefore, there are several improved weighting methods for weights of terms in the document vector. The well-known and commonly used one is the term frequency-inverse document frequency (TF-IDF) method where the importance of a term increases proportionally to the number of times the term appears in the document but is offset by the frequency of the term in the whole collection [11].

3. PRETO: PREPROCESSING TOOLKIT FOR TURKISH TEXTS

PRETO (Preprocessing Tool) is a standalone, platform-independent preprocessing tool developed specifically for preprocessing Turkish texts as well as English texts. PRETO offers several preprocessing options according to the language selection for the document collection. PRETO is designed and developed to be a powerful, high speed, efficient, and scalable text preprocessing tool with modest main memory consumption in order to preprocess very large document collections in a reasonable amount of time on a single processor. PRETO generates document-term matrices with different term weighting options in several sparse matrix file formats that are ready to use with common text mining tools.

3.1. PREPROCESSING OPTIONS

PRETO offers wide range of preprocessing options like stemming, stopword filtering, statistical term filtering, and n-gram generation. Processing steps of PRETO is shown in Figure 1.

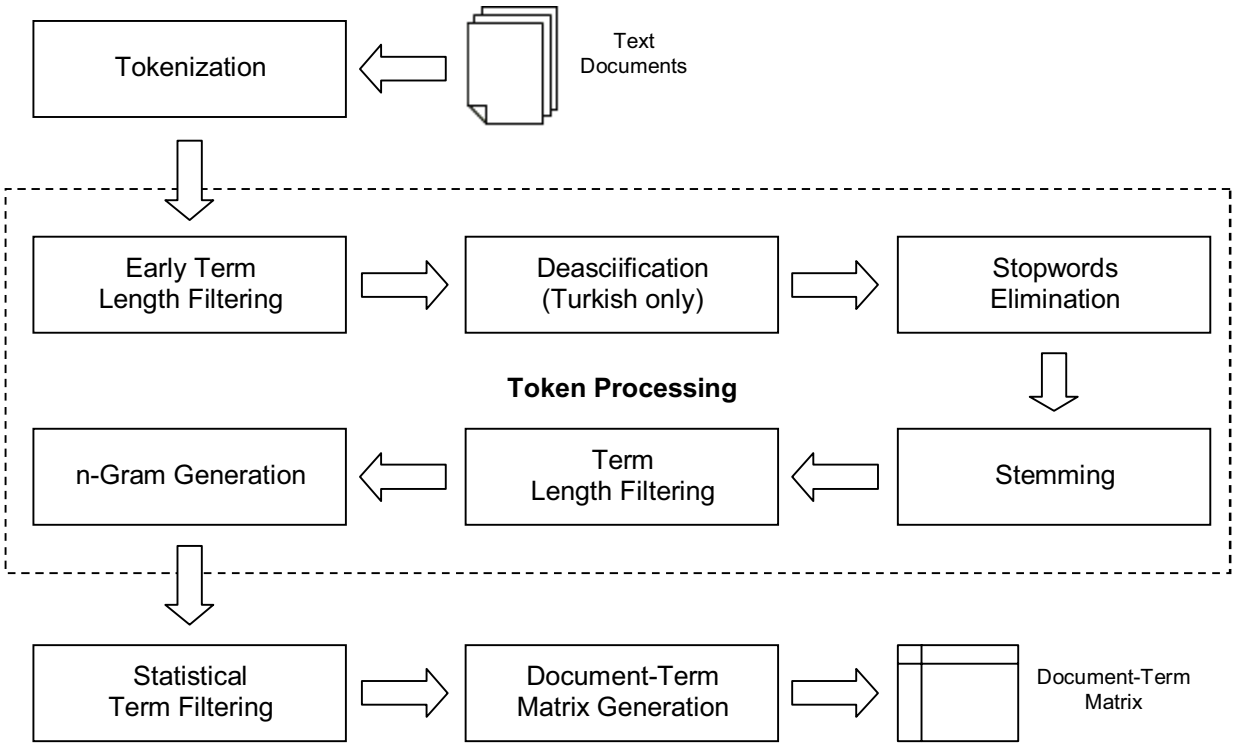

Figure 1. General view of the processing steps of PRETO.

PRETO can perform well-known Porter's algorithm for stemming texts in English [7]. For Turkish texts, on the other hand, there are two stemming options. Default Turkish stemmer is the affix stripping stemmer that reaches the stem of a word without using any lexicon while making the morphological analysis [3]. The other Turkish stemmer uses the root finding functions of Zemberek NLP library [1].

During our research, we often encounter documents written in Turkish but using non-Turkish keyboard schemes; for example, "cagrisim" instead of "çağrışım" (*connotation*), "isildama" instead of "ışıldama" (*shining*), etc. Therefore, a conversion called deasciification which restores accented characters from nonaccented Turkish text is usually necessary. We included a preprocessing option for deasciification using the related functions of Zemberek in PRETO.

PRETO has several term filtering options. First one is the stopword filtering that which terms are to be filtered out can be specified in a text file. Another filtering option filters the

terms according to their lenghts. For example, terms shorter than 4 charachters can be directly filtered out. Furthermore, PRETO offers options for statistical term filtering. Users can specify that a term must appear at least in a particular number of documents or at least in a particular percent of all documents. Default value of this option is 3 documents. Similarly, users can specify that a term can appear at most in a particular number of documents or at most in a particular percent of all documents. Default value of this option is 95%.

n-gram generation up to 5-grams is also available in PRETO. Users can make multiple selections among unigram to 5-gram options.

After all preprocessing is completed, PRETO generates document-term matrices with three different term weighting options in two common sparse matrix file formats. Weighting options available are term frequency only (TF), term frequency – inverse document frequency (TF-IDF), and normalized TF-IDF (TF-IDF-norm). Sparse matrix file formats available are Coordinate List (COO) file format and List of Lists (LIL) file format. In Coordinate List file format, each row of the file contains 3 numeric values: document ID, term ID, and term weight. In List of Lists file format, each row corresponds to a single document, and values in each row correspond to term IDs and term weights of the document. Many text mining tools make use of sparse matrices for fast and efficient operation. Both file formats are good for incremental matrix construction and supported by such tools. PRETO also generates text files that contain information about the documents processed and the terms that form the dictionary.

3.2. DESIGN AND IMPLEMENTATION

PRETO has an easy-to-use graphical user interface. It is developed in Java because Java applications are platform-independent; that is, PRETO can run virtually on any operating system that has Java Virtual Machine installed. In addition, Java offers implicit support for Unicode strings and files, which makes it possible to preprocess text files with any encoding scheme. Moreover, using Java allows us to use Zemberek NLP library for Turkish which is currently available in Java platform.

Because PRETO is designed and developed to be a high-performance, efficient, and scalable tool for preprocessing very large document collections, we paid very much attention to use most efficient data structures in its implementation. For example, stopword list is stored in a HashSet because it allows very fast existence test performed for each and every token. Similarly, for processing terms very quickly and efficiently, a HashMap is used for storing the global term dictionary. Since tokenization is carried out by reading a single character at a time from a text file, we used BufferedReader for reading characters efficiently from the underlying file stream.

Current implementation of PRETO does not provide support for preprocessing bilingual texts (i.e. text collections that consist of documents written in both Turkish and English, or documents that contain parts in both languages). Therefore, users need to specify the language of the text collection via the user interface.

Figure 2 displays the user interface of PRETO with a sample preprocessing performed. At the bottom of the screen, information about the preprocessing progress and the number of the file currently processed is displayed.

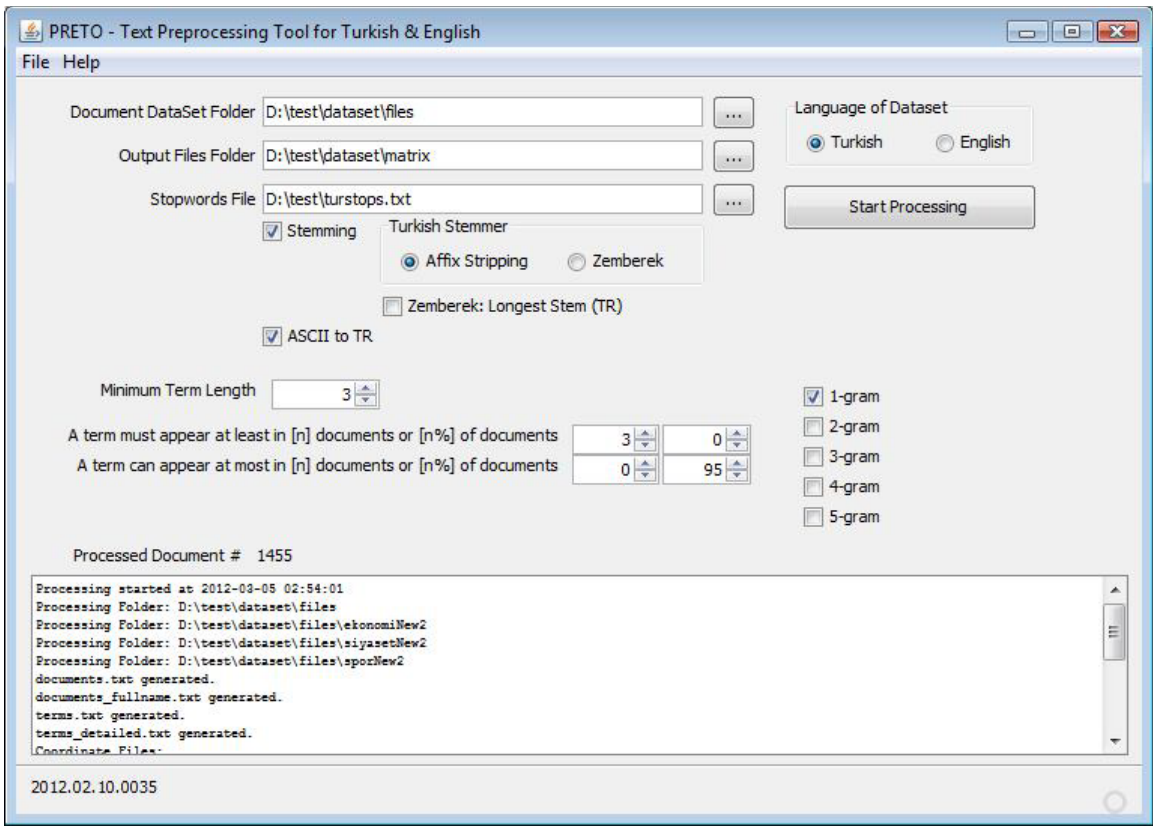

Figure 2. Graphical user interface of PRETO.

3.3. PERFORMANCE AND SCALABILITY MEASUREMENTS

In order to present the performance and scalability of PRETO with respect to size of document collection, we performed experiments on a Turkish document dataset.

We used PRETO to preprocess different number of Turkish text documents and measured the total processing time including the creation of document-term matrix files and other additional informational files. Besides processing time, we also measured the total memory consumption of PRETO for each run. Experiments were performed on a PC with Windows Vista Business platform running on a single core Mobile AMD Sempron 3600+ 2 GHz CPU and 2 GB main memory.

We ran PRETO with following options. We applied stopword filtering using a list of 182 words like "acaba", "bazı" and "daha". In addition, deasciification option was enabled. As the stemming option, affix stripping stemmer was used. After stemming, words shorter than 3 characters, words that appear in less than 3 documents, and words that appear more then 95% of the documents were filtered out using related options of PRETO. Among n-gram term generation options, only unigram option was selected.

Experimental results are shown in Table 1, including document collection statistics as total file size and average file size. In Table 1, processing time figures are in seconds, and memory usage figures are in megabytes. Figure 3 depicts these results.

Table 1. Experiment results of text preprocessing with PRETO.

Document Count	Total File Size (bytes)	Average File Size (bytes)	Preprocessing Time (seconds)	Memory Usage (megabytes)
1,000	1,583,663	1,583.66	25	140
2,500	4,933,604	1,973.44	69	153
5,000	9,353,193	1,870.64	136	172
10,000	19,177,649	1,917.76	265	206
15,000	27,532,235	1,835.48	382	237
20,000	35,621,920	1,781.10	504	266

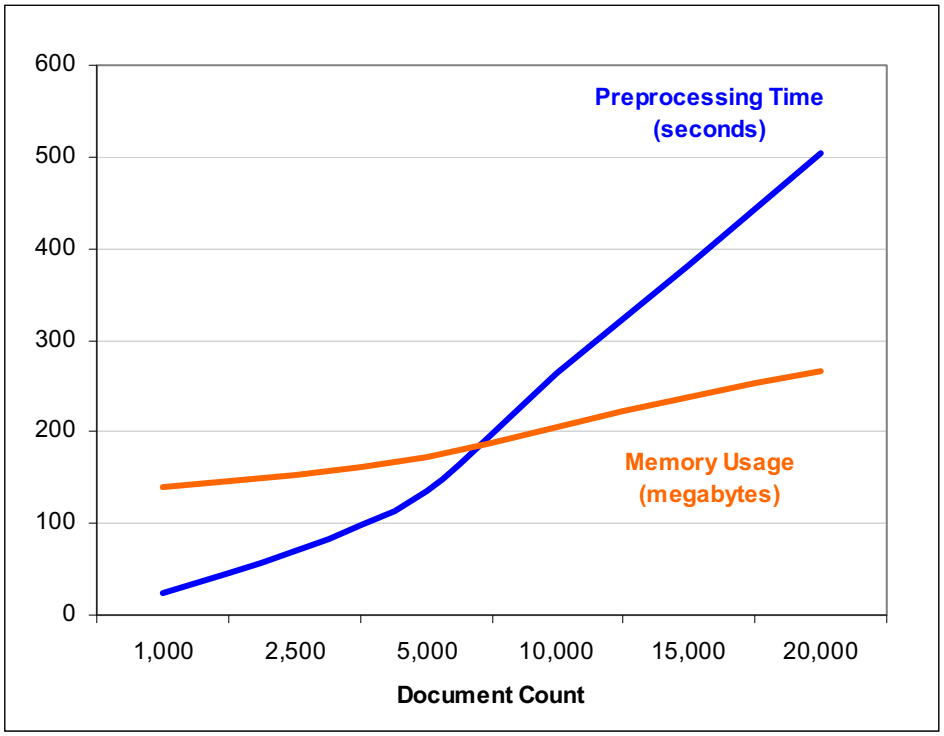

Figure 3. Preprocessing time and memory usage results of PRETO.

Experimental results show that time needed to preprocess Turkish text files of about the same size is proportional to the number of files. In addition, memory usage increases sublinearly with the number of files processed. Therefore, we can reliably say that PRETO is highly scalable in terms of CPU time and memory requirements and it is very suitable for preprocessing very large collections of Turkish documents for text mining purposes.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we present PRETO, a high-performance and scalable preprocessing tool developed specifically for preprocessing Turkish texts for text mining applications. PRETO has a broad range of preprocessing features like stemming, stopword filtering, statistical term filtering, and n-gram generation.

Our key motivation in the design of PRETO was to develop a high-performance, efficient, and scalable tool for preprocessing very large text collections especially in Turkish as well as English, and our experimental results support that PRETO meets our performance goals.

As a future work, we plan to add more stemming options for Turkish using the available techniques in the literature. Additionally, we plan to add multithreaded preprocessing capability for speed-up. PRETO will also be available as an open source project at Google Code for interested researchers and developers.

REFERENCES

[1] Akın, A. A. and Akın, M. D., "Zemberek, an open source NLP framework for Turkic Languages," 2007.

[2] Can, F., Koçberber, S., Balçık, E., Kaynak, C., Öcalan, H. Ç., and Vursavaş, O. M., "Information Retrieval on Turkish Texts," *Journal of the American Society for Information Science and Technology,* vol. 59, pp. 407-421, 2008.

[3] Eryiğit, G. and Adalı, E., "An Affix Stripping Morphological Analyzer for Turkish," in *International Conference on Artifical Intelligence and Applications*, Innsbruck, Austria, 2004, pp. 299-304.

[4] Feldman, R. and Sanger, J., *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*: Cambridge University Press, 2007.
[5] Han, J. and Kamber, M., *Data Mining: Concepts and Techniques*. USA: Morgan Kaufmann Publishers, 2006.
[6] Hotho, A., Nürnberger, A., and Paaß, G., "A Brief Survey of Text Mining," *LDV Forum-GLDV Journal for Computational Linguistics and Language Technology,* vol. 20, pp. 19-62, 2005.
[7] Porter, M. F., "An algorithm for suffix stripping," *Program,* pp. 130-137, 1980.
[8] Salton, G., Wong, A., and Yang, C. S., "A Vector Space Model for Automatic Indexing," *Communications of the ACM,* vol. 18, pp. 613-620, 1975.
[9] Tunalı, V., "Metin Madenciliği İçin İyileştirilmiş Bir Kümeleme Yapısının Tasarımı ve Uygulaması," Marmara Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul, Doktora Tezi, 2011.
[10] Weiss, S. M., Indurkhya, N., Zhang, T., and Damerau, F. J., *Text Mining: Predictive Methods for Analyzing Unstructured Information*. New York: Springer Science+Business Media Inc., 2005.
[11] Witten, I. H., Moffat, A., and Bell, T. C., *Managing Gigabytes: Compressing and Indexing Documents and Images*: Morgan Kaufmann Publishers, 1999.

ABOUT THE AUTHORS

Assist. Prof. Dr. Volkan Tunalı, Faculty of Engineering, Maltepe University, Phone: +90 216 626 10 50, E-mail: volkantunali@maltepe.edu.tr

Assist. Prof. Dr. Turgay Tugay Bilgin, Faculty of Engineering, Maltepe University, Phone: +90 216 626 10 50, E-mail: ttbilgin@maltepe.edu.tr