Analysis of Preprocessing Methods on Classification of Turkish Texts

Dilara Torunoğlu, Erhan Çakırman, Murat Can Ganiz, Selim Akyokuş and M. Zahid Gürbüz

Department of Computer Engineering
Doğuş University
Istanbul, Turkey
{dtorunoglu, ecakirman, mcganiz, sakyokus, mgurbuz}@dogus.edu.tr

Abstract— **Preprocessing is an important task and critical step in information retrieval and text mining. The objective of this study is to analyze the effect of preprocessing methods in text classification on Turkish texts. We compiled two large datasets from Turkish newspapers using a crawler. On these compiled data sets and using two additional datasets, we perform a detailed analysis of preprocessing methods such as stemming ,stopword filtering and word weighting for Turkish text classification on several different Turkish datasets. We report the results of extensive experiments.**

Keywords- Text Classification, Turkish Text Classification, Data preprocessing, stemming, stopword removal.

I. INTRODUCTION

Due to rapid developments in information technology and internet usage large amounts of textual data is accumulated in organizations. Merrill Lynch estimates that more than 85 percent of all business information exists as unstructured data – commonly appearing in e-mails, memos, notes from call centers and support operations, news, user groups, chats, reports, letters, surveys, white papers, marketing material, research, presentations and web pages [1]. This unstructured content includes valuable information that can be used for improving business communications, enhancing customer satisfaction and maintaining a competitive edge. Textual data mining is the process of extracting this valuable information from large amount of textual data. Textual data mining has several application areas such as customer relationship management, analysis of customer feedbacks and comments, email management, knowledge management in banks, detection of crime patterns and connections [2]. Text classification, a field of textual data mining, has great importance in processing and organizing large amounts of textual data automatically. Parallel to the trend in the world, Turkish textual data is also increasing rapidly. Turkish, as the native language of about 77 million people, is one of the most commonly used languages worldwide [3]. However, there are limited number of studies on Turkish text classification [4,5,6,7,8].

Generally, traditional classification algorithms from data mining field are used in text classification. These algorithms are primarily designed for structured data. At the current state, the main difference of text classification is the process of converting unstructured natural language text to the input format of these algorithms. This process is called preprocessing. Large portion of data preprocessing techniques in text mining stems from statistical natural language processing field. However, some recent work on literature show that some preprocessing methods such as stemming have very different affects on performance of information retrieval and text mining systems for different languages [9,10,11,12,13]. These results and major linguistic differences between English and Turkish language reveals the importance of an analysis and development of preprocessing methods for Turkish text classification.

Preprocessing is an important and critical step in text mining, because of the nature of the data we are working with. Text documents are unstructured and expressed in natural language which is extremely difficult to model. There are several methods used in text mining for preprocessing the text documents. Among these, tokenization, stopword filtering, stemming and word weighting are most commonly used [14,15,16]. These methods allow us to transform this unstructured data into structured format that the data mining algorithms can work on. They are also frequently used in information retrieval domain.

Most of the studies in information retrieval and text mining domain are applied to English texts. One of the few studies on Turkish language on information retrieval domain report that preprocessing methods such as stemming and stopword filtering have significant effect on the performance [9]. However, there is no similar study to observe the affect of preprocessing on text mining on Turkish texts. Motivated by this we analyze the effect of preprocessing in text classification on Turkish text using a wide range of datasets. We perform extensive experiments by using common preprocessing methods of stopword filtering, stemming, and word weighting, and report their effect on the performance of different types of classification algorithms.

This paper is organized as follows. Background and related work is discussed in Section 2. In Section 3, we give the details of preprocessing and classification methods we used in our experiment. Experimental setup and toolkit used are presented in Section 4. Results and a discussion of the effects of preprocessing in Turkish Text Classification are provided in Section 6. Section 7 includes concluding remarks and future work.

II. Background And Related work

Text classification is defined as labeling natural language texts documents with classes or categories from a predefined set [14]. Detailed analysis of text classification methods can be found in [14]. One of the distinguishing characters of text classification from other classification applications is the high dimensionality of the feature space. Generally features correspond to words or terms in text documents. This is called bag-of-words model. For benchmark datasets, number of features (also called dictionary size) can be tens of thousands.

Turkish is native language of over 77 million people [3] and belongs to the Altaic branch of the Ural-Altaic family of languages. The characteristic features of Turkish, such as vowel harmony, agglutination, and lack of grammatical gender, are universal within the Turkic family and the Altaic languages. The Turkish alphabet is consist of Latin characters and has 29 letters 8 of which are vowels and 21 are consonants. The letters in alphabetical order are; a, b, c, ç, d, e, f, g, ğ, h, ı, i, j, k, l, m, n, o, ö, p, r, s, ş, t, u, ü, v, y, and z. Seven of these (ç, ğ, ı, İ, ö, ş, ü) are specific to the Turkish and does not exists in English alphabet. On the other hand, q, w, x characters are specific to English and does not exists on Turkish alphabet. "Turkish is a free constituent order, an agglutinative language, and words are constructed using inflectional and derivation suffixes linked to a root" [9].

In one of important and recent works on information retrieval on Turkish text, the effect of different preprocessing methods is examined [9]. They report that stopword removal doesn't contribute a significant effect on Turkish text. They speculate that this may be related to tf*idf term weighting methods. One of important observations is that the use of stemming in preprocessing can increase effectiveness of the system up to 38%. They note that this drastic performance improvement is specific to the Turkish text. Motivated by these results, we examine the effect of preprocessing in Turkish text classification using similar preprocessing methods. There are limited studies on Turkish text classification. Among them, [4] uses a dataset consist of Turkish news. This dataset includes five different categories (economy, magazine, health, political, sports) and 230 documents for each category. They applied PCKimmo [17] toolkit for stemming, tf*idf term weighting and feature selection. They represent the words in a semantic space by using co-occurrence information and they report better performance results compared to bag-of-words model. A similar study [7] was done with a dataset that includes 18 categories and 630 documents in total, collected from news columns from Turkish newspapers. They use n-grams instead of bag-of-words model and report increase in the performance of classification. Similarly, in [18] they employ word n-grams instead of terms in classification and they applied unigram, bigram and trigram words. They concluded that the best results were achieved by using unigrams. In another study [5] on Turkish text, they proposed a spam filtering algorithm specific to Turkish language. They use a dataset of 750 e-mail messages which consist of 410 spam and 340 normal mail. They improved the performance of their system by incorporating language specific features. They exploit the fact that Turkish has much different structure and characteristics than English. Another study [6] was done with a larger dataset that consists of 13000 English and Turkish e-mail messages. They used a stemming algorithm specific to Turkish in their experiments and report that stemming has positive effect on the performance but it is not very significant. In another study [8], they analyzed the performance of classifiers when the longest or shortest roots are used. They used two different dataset, Milliyet newspaper and Wikipedia. They concluded that using only consonants in the roots gives better performance than using all letters in the roots.

In most of all these studies, research is conducted on relatively small datasets. The standard large size benchmark datasets such as 20 Newsgroups [19] or Reuters [20] do not exist for Turkish. Furthermore, to the best of our knowledge there is no study that systematically examines the effect of preprocessing on Turkish text classification.

III. Overview of Applied Methods

A. Preprocessing Methods

We use zemberek [21] and fixed prefix stemmer (FPS) [9] as stemming methods in our experiments. The fixed prefix stemming approach is a pseudo stemming technique which simply takes the first n characters of a given word. We experiment with different prefix lengths and report results for the first 3, 5 and 7 characters that we call FPS3, FPS5 and FPS7 respectively. Zemberek is an open source, platform independent, general purpose Natural Language Processing library and toolset designed for Turkic languages, especially for Turkish. Zemberek is officially used as spell checker in Open Office Turkish version and Turkish national Linux Distribution Pardus [21]. Zemberek is a dictionary based stemmer that uses a stem and suffix dictionary for analysis and stemming of words.

Term weighting is also an important part of preprocessing. In our experiments, we used two term weighting methods. The first one is binary weighting. In this weighting, terms are represented with binary values 0 or 1. The second one is term frequency (tf) weighting in which the terms are represented by their occurrence frequency in documents.

Stopword filtering is another commonly used preprocessing method in information retrieval and text classification. In stopword filtering, common frequent words (such as *and*, *or*, *this*) are removed from documents since they do not have any informative value for IR or text classification tasks. In [9], frequent 147 stopwords are determined with a semi-automatic generation approach by ranking all the words by their occurrences, selecting words that have higher frequencies than the predetermined threshold and then manually removing some common names. In our study, we use the same list as a stopword list in our experiments.

B. Classification Methods

We used the following classifiers in our study: Naïve Bayes, Naïve Bayes Multinomial, Support Vector Machines, And K-Nearest Neighbor.

Naive Bayes (NB) classifier is one of the widely used simple machine learning methods for text classification [14, 22]. Naïve Bayes assumes that the probability of each word occurred in a document is independent from the occurrence of other words in that document. Most commonly used event models in text classification are Multi-variate Bernoulli model and Multinomial model. In multi-variate Bernoulli model, a document is represented as a binary vector of terms taking value of 1 if the word exists in document and 0 if it does not. In contrast to multi-variate Bernoulli model, multinomial model a document as a vector of the word frequencies. Multinomial Naive Bayes (mnNB) assumes that the words in a document are drawn from an underlying multinomial distribution independently of each other [22]. A document is represented by the number of occurrences (or some weight) of words in the document.

The Support Vector Machine (SVM) is a relatively new classification method introduced in 1992 [23]. Although it is complex algorithm, SVM achieves high classification rates in many areas. SVM is basically a linear two-class classifier. Among the possible hyperplanes between two classes, SVM finds the optimal hyperplane between two classes by maximizing the margin among the closest points of classes. The points lying on the hyperplane boundaries are called support vectors. When two classes are not linearly separable, SVM projects data points into a higher dimensional space so that the data points become linearly separable by using kernel techniques. There are several kernels that can be used SVM algorithm. In our experiments we observed that linear kernel performs much better than the others.

The k-nearest neighbor (K-NN) classifier is a lazy learning instance-based method that does not include a training phase [24]. To classify an unknown document, K-NN algorithm identifies the k nearest neighbors in a given document space. K-NN algorithm uses a similarity function such as Euclidean distance or Cosine similarity to find neighbors. The algorithm ranks the nearest neighbors and predicts the class of the unknown document by computing the most frequent class label among the k nearest documents in the document space. The best choice of selecting the value of k depends upon the dataset or application. The implementation of K-NN algorithm is very easy, but it is computationally intensive, especially when the size of the training documents grows.

IV. Experimental setup

In order to examine the effects of several preprocessing methods, we use four different data sets that we call Milliyet_9c_1k, Hürriyet_6c_1k, 1150haber and Mini-newsgroups. The datasets Milliyet_9c_1k and Hürriyet_6c_1k are collected by using web crawler that we developed. The characteristics of each data set are explained below.

Milliyet_9c_1k dataset includes text from the columns of Turkish newspaper Milliyet[1] from years, 2002 to 2011. It is about 50 MB in size, contains nine categories and 1000 documents for each category. The categories of this dataset are café (cafe), Dünya (world), ege (region), ekonomi (economy), güncel (current), siyaset (politics), spor (sports), Türkiye (Turkey), yasam (life).

Hürriyet_6c_1k dataset includes news from 2010 to 2011 on Turkish newspaper Hürriyet[2]. It is about 24 MB in size, contains six categories and 1000 documents for each category. Categories in this dataset are Dünya (world), ekonomi (economy), güncel (current), spor (sports), siyaset (politics), yaşam (life).

1150haber dataset is obtained from a study done in [4]. It consists of 1150 Turkish news texts in 5 classes (economy, magazine, health, political, sports) and 230 documents for each category [4].

These three datasets are consists of Turkish text documents. Milliyet_9c_1k and 1150haber includes the writings of the column writers therefore they are longer and more formal. On the other hand Hurriyet_6c_1k datasets contains traditional news articles. They are more irregular and much shorter then documents of the other datasets. We further processed Milliyet_9c_1k dataset to remove specific words such as author names and email addresses that can uniquely identify specific class or document. In addition to these Turkish datasets, we also use an English dataset in order to observe and compare the affect of preprocessing methods on a different language. For this purpose we choose Mini-newsgroups dataset. Mini-newsgroups dataset is a subset of well-known 20-Newsgroups which is consists of 20000 messages taken from 20 Usenet newsgroups. This dataset includes 100 articles from each 20-newsgroup[3].The categories of this dataset are comp.graphics, misc.forsale, rec.autos, sci.space, talk.politics.mideast.

We use WEKA machine learning software package to run these algorithms [25]. Before starting experiments, all upper case letters were converted to lower case and UTF-8 were used for character encoding. A software framework is also developed in order to run experiments and analyze the results. The framework enables the execution of machine learning algorithms in WEKA on different datasets and test cases with several different parameters.

V. Results and Discussion

We have conducted extensive experiments by varying training set percentage and preprocessing parameters on several different datasets of two different languages. We use four different datasets, three different preprocessing methods and three different classification algorithms in order to determine the effects of different preprocessing methods on Turkish text. As explained before, preprocessing methods

[1] www.milliyet.com.tr
[2] www.hurriyet.com.tr
[3] Http://People.Csail.Mit.Edu/Jrennie/20newsgroups/

include stopword removal (STREM), Fixed Prefix Stemmer (FPS) and zemberek (ZEM). FPS takes the first n characters of a given word. We carry out experiments with the FPS stemmer by using different prefix lengths and report results for three, five and seven characters called as FPS3, FPS5 and FPS7 respectively. For the English dataset we used lovins (LOV) stemmer [26] that is available in WEKA. In experiments, we used the following classifiers: Naïve Bayes (NB), multinomial Naïve Bayes classifier (mnNB), k-nearest neighbor classifier (K-NN) and support vector machine (SVM). The number of nearest neighbors is selected as one, five and thirty. Among these five neighbors gives better accuracies yet still very low compare to other classifiers.

The characteristics of each dataset after preprocessing are shown in tables 1, 2, 3 and 4. The first row (titled None) gives statistics without preprocessing. In tables, STREM means that stopword removal is applied. FPS means that fixed prefix stemmer is applied. For example, STREM+FPS5 mean that both stopword removal and fixed prefix stemming with first five characters are applied; STREM+ZEM mean that stopword removal and zemberek algorithm are applied. We find that FPS5 results better performance than FPS3 and FPS7 in general. This is also consistent with the observation of [9]. As noted in [9] Turkish words are not much longer than seven characters and their roots are between 3 to 7 characters. The total number of terms statistics in these tables show that the FPS3 stemmer results a drastic reduction in dictionary size. For instance, in 1150haber dataset dictionary size reduced to 2457 from 11040. As a result, it performs poorly compared to other stemmers because it only takes the first three characters, and three characters are not informative enough for classification. On the other hand, since average term length is around seven in our Turkish dataset FPS7 stemmer does not provide considerable performance differences. There is one interesting observation about ZEM stemmer although, it has similar average term length with other stemmers it causes a greater reduction in average number of terms per document. We speculate that ZEM is making better clustering of words.

We perform experiments with several different classifiers including k-nearest neighborhood, NB, mnNB, SVM (with different kernels) and two different weighting schemes: binary and term frequency. We found that term frequency scheme produce better results compare to binary and mnNB, and SVM with linear kernel perform much better than others classifiers. Consequently we only report the results of these two classifiers with term frequency weighting in the figures.

TABLE I. MILLIYET_9C_1K CORPUS (9000 DOCUMENTS)

Preprocessing	# of terms	Avg. # of terms per document	Avg. term length
None	63371	317,51	8,45
STREM	63240	285,61	8,46
STREM +FPS3	4852	214,09	2,92
STREM +FPS5	20620	270,15	4,68
STREM + FPS7	39122	291,43	6,18
STREM +ZEM	17053	224,62	6,26

TABLE II. HÜRRIYET_6C_1K CORPUS (6000 DOCUMENTS)

Preprocessing	# of terms	Avg. # of terms per document	Avg. term length
None	18280	124,61	6,68
STREM	18154	100,41	6,70
STREM +FPS3	3330	94,09	2,88
STREM +FPS5	11161	101,53	4,50
STREM + FPS7	15669	102,18	5,65
STREM +ZEM	11291	93,56	5,78

TABLE III. 1150HABER CORPUS (1150 DOCUMENTS)

Preprocessing	# of terms	Avg. # of terms per document	Avg. term length
None	11040	127,41	7,30
STREM	10860	107,06	7,35
STREM +FPS3	2457	110,66	2,91
STREM +FPS5	6583	120,00	4,66
STREM + FPS7	9589	119,20	6,01
STREM +ZEM	5092	107,67	5,61

TABLE IV. MINI-NEWSGROUPS CORPUS (2000 DOCUMENTS)

Preprocessing	# of terms	Avg. # of terms per document	Avg. term length
None	13943	180,77	6,12
STREM	13468	120.86	6,16
STREM +FPS3	5065	132.24	2,88
STREM + FPS5	10940	128.92	4,39
STREM +LOV	10621	131.70	5,26

To evaluate the performance of classifiers, we use the repeated holdout method. The standard holdout method involves splitting the dataset into two random disjoint (training and test) subsets with a training percentage *p*. In the repeated holdout method, the holdout procedure is repeated *r* times in order to minimize any bias arising from a particular sample chosen for holdout. In this method, the overall accuracy rate is an average of accuracies of *r* repetitions. In text mining applications, the repeated holdout evaluation method is usually preferred over the standard holdout method since it produces a more reliable estimate.

In our experiments, the percentages of training subsets, *p*, are selected as 1%, 3%, 5%, 10%, 30%, 50%. For the repeated holdout evaluation, the number of repetitions, *r*, is chosen as 10. We have performed 10 runs for each of the training percentages. In Figures 1-8, each bar represents the average accuracy of these ten runs. Error bars at the top of each bar shows standard deviations of these ten runs. As expected, the error bars shows high deviation between the results of these ten runs when the training set size is small (e.g. 1% and 3%). As the training set size increases, the deviation decreases and the results become more stable.

Figure 1-8 shows performance of mnNB and SVM classifiers trained with different training set sizes that are preprocessed with stop word filtering (STREM) and two different stemmers (FPS5 and ZEMBEREK). In order to compare performance of the preprocessing settings on both English and Turkish language, Figure 7 and 8 includes the accuracies for mini-newsgroups dataset which consist of English documents.

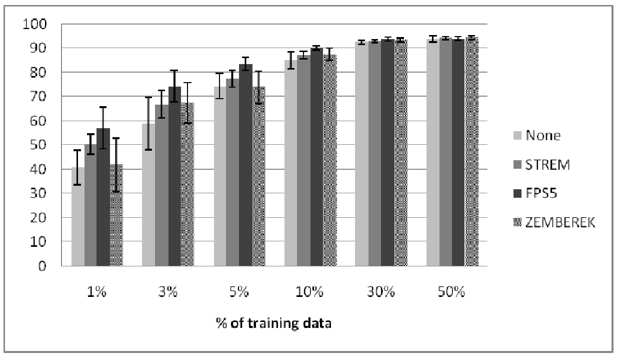

Figure 1. Affect of preprocessing in mnNB with 1150haber

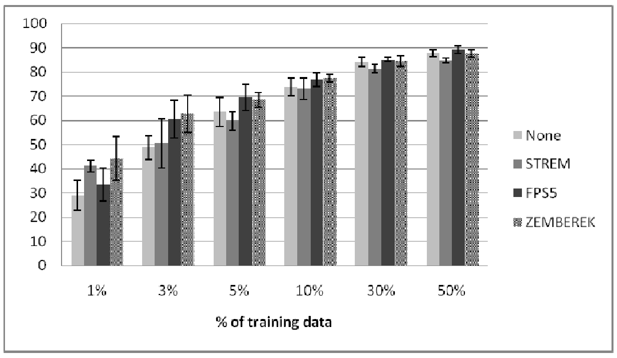

Figure 2. Affect of stemming in SVM with 1150haber

The effect of stopword removal and stemming is only visible on small training set sizes. We speculate that the reason for this is when the training data is small; data is highly sparse and stemming helps in reducing the sparsity by grouping semantically similar words together. On these small training set sizes, zemberek seems to be the best stemmer for SVM algorithm. On the other hand, mnNB classifier seems to perform better with STREM alone and with FPS5 stemmer combined with STREM. However, given the large standard deviations of accuracies in small training sets we can't confidently state these differences are significant. When the training set size increase, data becomes less sparse and there will be more information for classifiers to exploit. As a result we do not see important differences of applying stopword filtering and stemming.

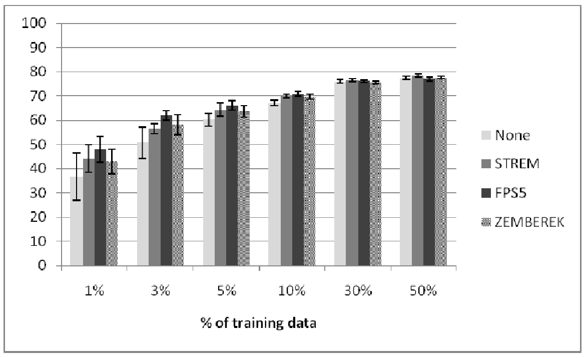

Figure 3. Affect of stemming in mnNB with Hürriyet_6c_1k

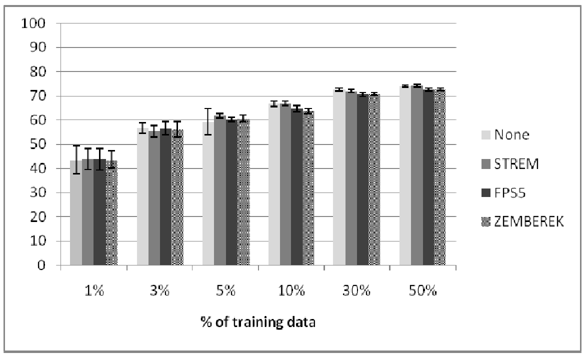

Figure 4. Affect of stemming in SVM with Hürriyet_6c_1k

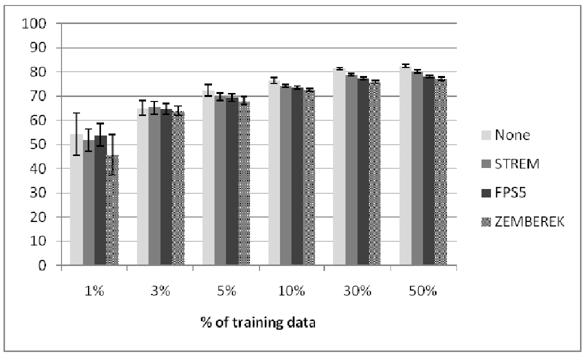

Figure 5. Affect of stemming in mnNB with Milliyet_9c_1k

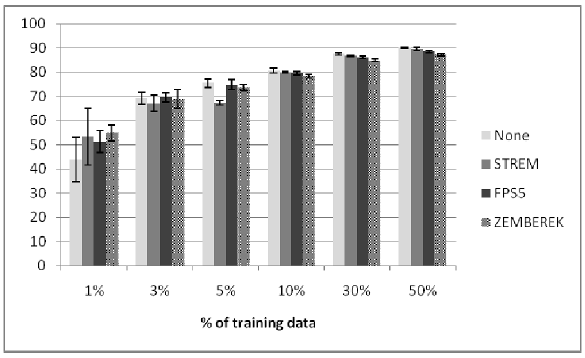

Figure 6. Affect of stemming in SVM with Milliyet_9c_1k

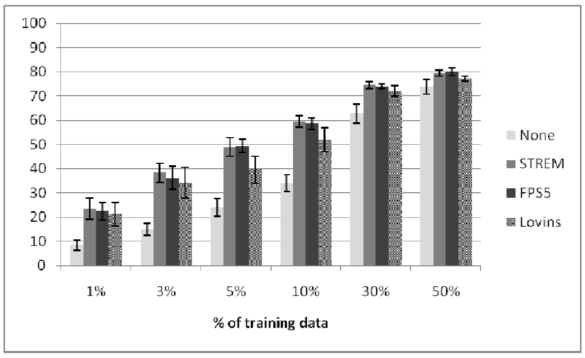

Figure 7. Affect of stemming in mnNB with Mini-newsgroups

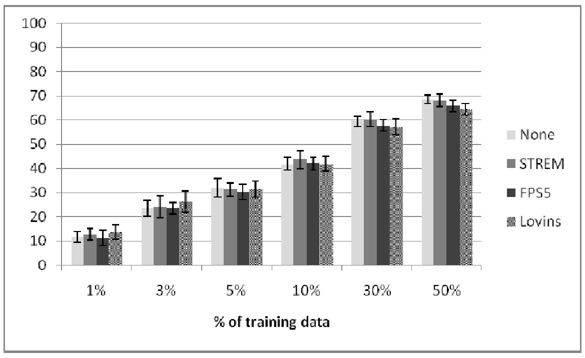

Figure 8. Affect of stemming in SVM with Mini-newsgroups

VI. CONCLUSION AND FUTURE WORK

The objective of this study is to analyze the effect of preprocessing methods in text classification on Turkish texts. We compiled two large datasets from Turkish newspapers using a crawler. We perform extensive experiments by using common preprocessing methods and report their effect on the performance of different types of classification algorithms.

Based on the observations above, we conclude that stemming and stop word filtering has very little impact, if any, on the overall accuracies of two of the most commonly used text classification algorithms on a wide variety of Turkish datasets. It is reported that stemming has significant impact on Turkish information retrieval (IR) [9]. This result may due to the agglutinative property of Turkish language. Applying stemming increases recall rates in IR. However, new words formed by adding suffixes to roots of Turkish words may result in complete change in the value of the word for classification. As a result, classification accuracies are not affected significantly by preprocessing and may actually drop on some cases.

As a future work, we plan to apply feature selection methods and weighting methods such as tf*idf on Turkish documents and observe their affect on performance. We also consider developing better stemming and word weighting methods that exploit semantics of Turkish words for classification.

REFERENCES

[1] R. Blumberg, S. Atre, "The Problem With Unstructured Data", Information Management Magazine, February 2003

[2] A. Zanasi, "Text Mining And Its Applications To Intelligence", Crm And Knowledge Management (Advances In Management Information). Wit Press. (2005).

[3] European Commission, "Europeans And Their Languages (Survey)" Special Eurobarometer 243, 2006, Retrieved 2011-02-14.

[4] M.F. Amasyalı, A. Beken, "Türkçe Kelimelerin Anlamsal Benzerliklerinin Ölçülmesi Ve Metin Sınıflandırmada Kullanılması", Siu 2009, Antalya.

[5] L. Özgür, T. Güngör, F. Gürgen, "Adaptive Anti-Spam Filtering For Agglutinative Languages: A Special Case For Turkish", Pattern Recognition Letters, Vol. 25, Pp. 1819–1831 (2004).

[6] A. Çiltik, T. Güngör, "Time-Efficient Spam E-Mail Filtering Using N-Gram Models", Pattern Recognition Letters, Vol. 29, Pp.19–33, (2008).

[7] M.F. Amasyali, B. Diri, "Automatic Turkish Text Categorization In Terms Of Author, Genre And Gender" , 11th International Conference On Applications Of Natural Language To Information Systems-Nldb2006, Austria,(2006).

[8] Z. Çataltepe, Y. Turan, F. Kesgin, "Turkish Document Classification Using Shorter Roots", Signal Processing And Communication Applications, 2007, Siu 2007, Ieee 15a

[9] F. Can, S. Kocberber, E. Balcik, C. Kaynak, H. C. Ocalan, O. M. Vursavas, "Information Retrieval On Turkish Texts", Journal Of The American Society For Information Science And Technology. Vol. 59, No. 3 (February 2008), Pp. 407-421

[10] D. Harman, "How Effective Is Suffixing?" Journal Of The American Society For Information Science, 1991,42(1),Pp 7-15

[11] P. Ahlgren, & J. Kekalainen,. "Indexing Strategies For Swedish Full Text Retrieval Underdifferent User Scenarios", Information Processing And Management, 2007,43(1),Pp 81-102

[12] S. Larkey, L. Ballesteros, & M. Connell, "Improving Stemming For Arabic Information Retrieval: Light Stemming And Co-Occurrence Analysis", In Proceedings Of The 25th International Conference On Research And Development In Information Retrieval (Acm Sigir' 02) (2002) (Pp. 275- 282). Tampere: Acm.

[13] R.M. Duwairi, M.N. Al-Refai, N. Khasawneh, "Feature Reduction Techniques For Arabic Text Categorization", Journal Of The American Society For Information Science And Technology, 60(11), Pp. 2347–2352, 2009.

[14] F. Sebastiani, "Machine Learning In Automated Text Categorization", Acm Comput. Surv., 34(1), Pp.1-47, Mar. 2002.

[15] A. Aizawa, "Linguistic Techniques To Improve The Performance Of Automatic Text Categorization", Proceedings Of Nlprs-01, 2001

[16] Y. Yang, "An Evaluation Of Statistical Approaches To Text Categorization" Information Retrieval, 1999.

[17] K. Oflazer, "Two-Level Description Of Turkish Morphology. Literary And Linguistic Computing", 9(2), Pp. 137-148, (1994).

[18] A. Güran, S. Akyokuş, N. Güler, Z. Gürbüz, "Turkish Text Categorization Using N-Gram Words", Inista 2009, Trabzon.

[19] K.Lang, "Newsweeder: Learning To Filter Netnews" , Proceedings Of The 12th International Machine Learning Conference, 1995.

[20] T. Rose, M. Stevenson, M.Whitehead, "The Reuters Corpus Volume 1- From Yesterday's News To Tomorrow's Language Resources".

[21] A.A Akın , M.D. Akın, "Zemberek, An Open Source Nlp Framework For Turkic Languages".

[22] A. Mccallum, K. Nigam: "A Comparison Of Event Models For Naive Bayes Text Classification", In: Aaai-98 Workshop On 'Learning For Text Categorization', 1998.

[23] J. Platt: Fast Training Of Support Vector Machines Using Sequential Minimal Optimization. In B. Schoelkopf And C. Burges And A. Smola, Editors, Advances In Kernel Methods - Support Vector Learning, 1998

[24] D.W. Aha, D. Kibler, And M.K. Albert, "Instancebased Learning Algorithms.", Machine Learning, 6 , Pp. 37-66, 1991.

[25] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools And Techniques, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[26] J.B, Lovins, "Development Of A Stemmer Algorithm", Mechanical Translation And Computational Linguistics, 11,1986,Pp 22-31