Turkish Text Categorization Using N-Gram Words

Aysun Güran
Yıldız Technical University,
aysunguran@gmail.com

Selim Akyokuş
Doğuş University,
sakyokus@dogus.edu.tr

Nilgün Güler Bayazıt
Yıldız Technical University,
guler@yildiz.edu.tr

M.Zahid Gürbüz
Yıldız Technical University,
mgurbuz@gmail.com

Abstract

An N-gram is a representation method that consists of a sequence of N-contiguous characters or words. There have been so many studies which use N-gram based representations for the traditional text classification tasks. In contrast to other languages, the studies in Turkish are limited. In this paper, we analyze text classification algorithms on a Turkish dataset by using N-gram words. We have compared several classifiers (Bayesian probabilistic classifiers, nearest neighbor classifiers and decision trees) using different types of features. We applied the classifiers on different data sets that are represented with unigram, bigram and trigram words. In the experiments, a total of 600 text documents that are assigned to six categories were used and the best success rate of 95.83% was achieved by using unigrams.

1. Introduction

The World Wide Web is a huge source for storing and accessing linguistic documents. The amount of linguistic documents in the World Wide Web grows rapidly and this rapid growth has raised the great interest in helping people for finding easier ways to organize and classify these resources. Automatic text categorization or text classification methods enable the organization or categorization of a set of documents into different categories or classes. Many classification problems have been solved manually by the use of some rules commonly written by hand. But creating these rules is labor-intensive. Instead of using hand-written rules, the text categorization approaches uses machine learning methods to learn automatic classification rules based on human-labeled documents. It is obvious that labeling is an easier task than writing rules. Hence, text categorization can be considered as an effective method for automatic assignment of documents to the predefined categories according to their context.

A number of classification and machine learning techniques have been applied to text categorization. These techniques include Bayesian probabilistic classifiers, nearest neighbor classifiers and decision trees. Some of these techniques are based on N-grams.

In contrast to the other languages, text categorization (TC) hasn't been studied so much in Turkish language. The work in Turkish TC is very limited. Amasyalı and Yıldırım developed a system for TC and they achieved 76% success ratio [1]. Amasyalı and Diri used character N-grams and evaluated some classification algorithms for determining the author of the text, genre of the text and gender of the author [2]. The success in these problems was obtained as 83%, 93% and 96%, respectively. Yılmaz, Gençtav, Usta et.al., proposed a new feature extraction method for TC [3]. Using the new method they achieved 96.25% success with the Naive Bayes algorithm. Güven, Bozkurt, and Kalıpsız [4] applied Latent Semantic Analysis method on N-gram word documents. Özgür, Güngör and Gürgen [5] developed anti-spam filtering methods for Turkish and for agglutinative languages in general. They obtained a success rate of 90% based on Artificial Neural Networks (ANN) and Bayesian Networks (BN) algorithms.

In this paper, we applied several classifications methods on a Turkish data set that are represented by unigram, bigram and trigram words. We usually obtained high classification rates. The best results are 95.83%, 93.17%, 52.83%, respectively.

This paper is organized as follows: Section 2 describes TC and summarizes the TC algorithms used in this paper. Section 3 gives a brief description of data collection and representation. Section 4 explains the preprocessing phase. Section 5 outlines the evaluations and results. Finally Section 6 gives the conclusion and suggested future work directions.

2. Text categorization

Automatic text categorization (also known as text classification-TC) is the process of identifying the class which a document belongs to. In TC, there is a document space $D = \{d_1, d_2, .., d_n\}$, where each document d_i is represented by a |V| dimensional vector $d_i = \{w_1, w_2, .., w_{|V|}\}$, where each w_k is some

weight of term k in document d_i and a fixed predefined set of classes $C = \{c_1, c_2, .., c_{|c|}\}$ which is also called categories. The categories are set by using human-labeled documents. The goal of TC is to build a classification function Υ that maps documents to their classes:

$$\Upsilon : \mathrm{D} \rightarrow \mathrm{C} \quad (1)$$

There are many types of algorithms and/or methods used in text categorization [6]. In this study, we use the following classifiers after preprocessing step: Naïve Bayes, Complementary Naïve Bayes, Naïve Bayes Multinomial, J48 (C4.5 Decision Tree), K-Nearest Neighbor. We use WEKA [7] machine learning software package to run these algorithms. To obtain better classification accuracy we included feature selection (Information Gain measure) in our experiments. The classification methods used in this study are briefly reviewed below:

2.1. Naive bayes

The Naive Bayes (NB) classifier is a widely used machine learning technique for text categorization because it performs well in many domains, despite its simplicity [8]. Using Bayes' rule, the model is inverted in order to predict the most likely class for a new document. The most likely class C_{NB} for document $d_q = \{\mathrm{w}_1, \mathrm{w}_2, .., \mathrm{w}_{|V|}\}$ is computed using

$$C_{NB} = \mathrm{argmax}_{c_j \in C}\, p(c_j) p(\mathrm{w}_1, \mathrm{w}_2, .., \mathrm{w}_{|V|} | c_j) \quad (2)$$

In Naive Bayes, the attributes are assumed to be independent, then

$$C_{NB} = \mathrm{argmax}_{c_j \in C}\, p(c_j) \prod_{i=1}^{|V|} p(\mathrm{w}_i | c_j) \quad (3)$$

where $p(\mathrm{w}_i | c_j)$ and $p(c_j)$ are estimated from training documents with known classes. $p(c_j)$ *is* the probability of a class c_j in training documents. Given a class c_j, $p(\mathrm{w}_i | c_j)$ *is* the conditional probability of word w_i of a query document d_q in training documents.

2.2. Complement naive bayes

The Naive Bayes classifier has two systemic problems: a) skewed data bias that occurs when there are more training examples for one class than another and b) weight magnitude errors caused by assumption of independence of attributes. To overcome these problems, a new version of Naïve Bayes called Complement Naive Bayes (CNB) is developed [9]. In CNB, the most likely class C_{CNB} for document $d_q = \{w_1, w_2, .., w_{|V|}\}$ is computed using

$$C_{CNB} = \mathrm{argmax}_{c_j \in C} \log p(c_j) - \sum_{i=1}^{|V|} \mathrm{f_i} \log p(\mathrm{w}_i | \bar{c_j}) \quad (4)$$

where $p(\mathrm{w}_i | \bar{c_j})$ is estimated by the frequency of the term $\mathrm{w_i}$ occurred in classes other than c_j (i.e. its complement). $\mathrm{f_i}$ is the frequency count of word i in a document. This approach is particularly suited when only few labeled examples are available for each class c_j.

2.3. Multinominal naive bayes

In Multinomial Naive Bayes (MNB), the words in a document are assumed to be drawn from an underlying multinomial distribution independently of each other. A document is represented by the number of occurrences (or some weight) of words in the document. In MNB, the most likely class C_{MNB} for document $d_q = \{\mathrm{w}_1, \mathrm{w}_2, .., \mathrm{w}_{|V|}\}$ is computed using

$$C_{MNB} = \underset{c_j \in C}{\mathrm{argmax}} \log p(c_j) + \sum_{i=1}^{|V|} \mathrm{f_i} \log p(\mathrm{w}_i | c_j) \quad (5)$$

where the conditional probability $p(\mathrm{w}_i | c_j)$ is the relative frequency of term $\mathrm{w_i}$ in documents belonging to class c_j. It has been shown by McCallum and Nigam [10] that the Multinomial Naive Bayes classifier based on the multinomial distribution outperforms the multi-variate Bernoulli classifier on text classification applications. The multinomial Naive Bayes model is particularly appropriate for text classification.

2.4. C4.5 decision tree (J48)

J48 is an implementation of the Quinlan's [11] C4.5 algorithm. The C4.5 algorithm builds a decision tree model using a set of training data. The algorithm uses the greedy strategy to induce decision trees for classification. The C4.5 is a recursive algorithm that uses *information gain ratio* measure to select an attribute that splits the data set into smaller data subsets. The attribute with highest information gain ratio is selected as a splitting attribute. The algorithm recursively runs on smaller subsets to form a decision tree by finding the remaining splitting attributes at each step. The set of attributes found at each step constitutes the decision tree. Decision trees is probably the most widely used machine learning method in practice to date. Decision trees is also a popular classification method used in text classification.

2.5. K-nearest neighbor

The K-nearest neighbor (K-NN) classifier is a lazy learning instance-based method that does not include a training phase [12]. To classify an unknown document d_q, K-NN algorithm identifies the k nearest neighbors in a given document space $D = \{d_1, d_2, .., d_n\}$. 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 data set 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.

3. Data collection and representation

We ran algorithms on a dataset collected from the Web. The dataset consists of 600 text documents that are assigned to six categories (auto, politics, medicine, magazine, economy and sports). Each category has 100 documents. We considered the documents as a set of lines with one unigrams, bigrams and trigrams separated with a pipe, as shown in Figure 1:

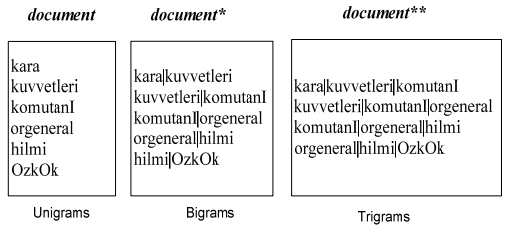

Figure 1. Unigram, bigram, and trigram document representations

We used the vector space model for representing text documents in our corpus. In the vector space model, the text documents are captured as a vector where each component is related to a word which is assigned to a value associated with some weight in the vocabulary[1]. This weight was computed using the TFIDF measure. Consider the term frequency (*freq(d,t)*) as the number of occurrences of term t in the document d. The term frequency matrix (*TF(d,t)*) represents the association between the word t and document d. Inverse document frequency (IDF) models the importance of a term. If a term appears in many documents, its importance will be reduced according to its reduced discriminative power. We used the Cornell Smart [13] system to compute the TF and IDF measures:

$$TF(d,f) = \begin{cases} 0, & if\ freq(d,t) = 0 \\ 1 + \log(1 + \log(freq(d,t))), & otherwise \end{cases} \quad (6)$$

$$IDF(t) = log\frac{1+|d|}{|d_t|} \quad (7)$$

In the Eq.7, d is the total number of the documents in the corpus and d_t is the number of documents containing term t.

In the vector space model, TF and IDF are used together.The complete model that forms the TFIDF measure is shown in the following equation:

$$TFIDF(d,t) = TF(d,t) * IDF(t) \quad (8)$$

4. Document preprocessing

Turkish has agglutinative morphology with productive inflectional and derivational suffixations [14]. The number of word forms one can derive from a Turkish root may be in the millions [15]. Hence the dimension of the vector space becomes very large. Stemming can be used as a method to reduce the number of different word forms. A stemming algorithm reduces all inflected forms of a word to the same stem. In this paper, we used the two-level morphological parser (PC-KIMMO) to parse the words to obtain the roots [14]. Figure 2 shows an overview of two-level model functions.

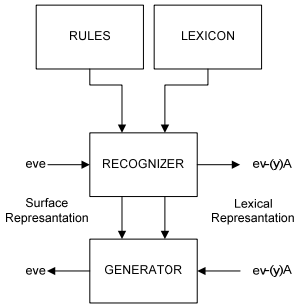

Figure 2. Main components of PC-KIMMO

In this model, a word is represented as a direct, letter-for-letter correspondence between its lexical and surface form [16]. For example, the word "*kitabım"* is given this two-level representation:
Lexical form: k i t a p + I m[2]
Surface form: k i t a b 0 ı m[3]

We applied four preprocessing phases to our dataset. For the first preprocessing phase normal texts were written with lower case letters in order to use PC-KIMMO. Some upper cases letters were used to indicate the special Turkish characters:

$\{C \rightarrow ç,\ \ S \rightarrow ş,\ \ I \rightarrow ı,\ \ O \rightarrow ö,\ \ U \rightarrow ü,\ \ G \rightarrow ğ\}$

For the second preprocessing phase, we created a stop list dictionary from the lexicons (*baglac.lex* and *pronouns.lex*) used in PC-KIMMO. "*baglac.lex*" contains connectives, exclamations, questions and some adverbs and "*pronouns.lex"* contains pronouns in Turkish. Stop words are those words that are too frequent in our dataset, and they are considered redundant words for the categorization task. We removed stop words, digits, single Turkish letters (letters that don't belong to a word) and punctuations.

[1] Vocabulary is a vector that consists all different words from all documents in the corpus and its size is considered as the vector space dimension.

[2] + is a morpheme boundary.

[3] 0 is a null character.

For the third phase we applied PC-KIMMO to get the root of a word from the output of the morphological parser.

For the fourth phase we selected words by their frequency distributions among documents by using Information Gain (IG) measure. This is used to select features with the highest discrimination.

5. Evaluation

In first part of our experiments, we made analysis on a document set with unigram words (single words). The training set is represented as a vector of single words. In these experiments, we analyzed both performance of classification algorithms and the effect of removing stop words in Turkish. These experiments include two data sets: 1) original data set including stop words and 2) reduced data set after removal of stop words. The original data set has a feature space of 30963 words. The reduced data set has a feature space of 30653 after the removal of stop words. The classification algorithms CNB, MNB, NB, KNN and J48 are run on these data sets using 10-fold cross-validation. As shown in Figure 3, using K-NN we have made experiments on these data sets to get the best choice of selecting the value of K. We selected K=7 as a best choice for both the original and the reduced data sets.

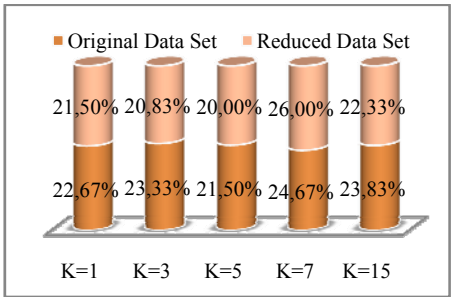

Figure 3. Performance of K-NN according to different K values

Table 1 shows the classification accuracy of each of the classifiers. The classification accuracy is defined as the percent of test documents are correctly classified by a classifier. As it can be seen from Table 1, both data sets produce similar classification accuracies. It means that the removal of stop words does not affect the accuracies of the classifiers, but reduce the feature space.

Table 1. Analysis of performance of classifiers and the effect of removal of stop words

Algorithms	The original data set without stop words removal (30963 words)	The reduced data set after stop words removal (30653 words)
CNB	94.67%	94.67%
MNB	93.67%	93.67%
NB	85.83%	86.33%
J48	68%	66.83%
KNN	24.67%	26%

In the second part of the experiments, the data sets are first preprocessed by removing stop words and stemming. In order to apply K-NN algorithm, we tried to determine the best value of K. Figure 4 shows the optimal K values for K-NN algorithm on the stemmed unigram data set and the reduced data set with IG feature reduction algorithm. We obtained K=7 for the stemmed data set and K=3 for the reduced data set.

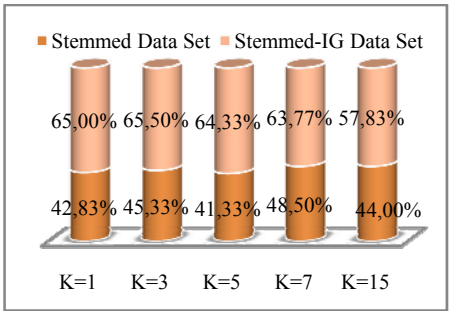

Figure 4. Performance of K-NN according to different K values

Table 2 shows the results of experiments done on unigram, bigram and tri-gram words. In these experiments, we have three training sets which are represented as a vector of single words (unigram), a vector of a pair of adjacent words (bigrams), and a vector of three consecutive words (trigrams) respectively. Three data sets have a feature space of 10192, 84381 and 109584 words respectively. As it can be seen form Table 2, the unigram representation has better classification accuracies compared with bigram and trigram representations. On the unigram representation, the MNB algorithm produces the best accuracy with a rate of %94.67. On the bigram representation, the CNB algorithm produces the best accuracy with a rate of %93.17. The performance of classifiers degrades considerably on the trigram representation.

Table 2. Analysis of Performance of Classifiers on unigram, bigram, and trigram representations.

Algorithms	Unigram Representation (10192 words)	Bigram Representation (84381 words)	Trigram Representation (109584 words)
CNB	94%	93.17%	50.83%
MNB	94.67%	86.33%	30.17%
NB	85.5%	81.83%	52.83%
J48	73.5%	60%	23.33%
KNN	48.5%	18.33%	18.5%

Table 3 shows the performance of classifiers on reduced data sets with unigram, bigram, and trigram representations. On these data sets, the feature space is reduced by using IG feature selection algorithm. We used the information gain measure in order to rank the features and we selected the features whose

rank values are greater than zero. After feature selection, the three reduced data sets have a feature space of 806, 526 and 111 words respectively. As it can be seen from Table 3, the reduction of feature space does not affect the classification accuracy of the most of the classifiers significantly, but improves the running time.

Table 3. Analysis of Performance of Classifiers on unigram, bigram, and trigram representations on reduced data sets with feature selection.

Algorithms	Unigram Representation (806 words)	Bigram Representation (526 words)	Tri-gram Representation (111 words)
CNB	94.17%	81.17%	41.33%
MNB	95.83%	81.67%	41%
NB	93.33%	80.17%	43%
J48	75.33%	61%	22%
KNN	65.5%	44.33%	38.33%

6. Conclusions and future works

In this paper, we analyze text classification algorithms on a Turkish dataset by using N-gram words. From the evaluation results, it can be said that the K-NN classifier shows the worst performance among the other algorithms. Large feature space has a very negative impact on it. The K-NN classifier shows better performance if the size of the vector space decreases. The use of bigram and trigram representations has a negative influence on the accuracy rates of K-NN classifier. The best performance (65.5%) for K-NN is obtained when the unigram representation is used as shown in Table 3.

The J48 classifier has the highest performance values on the reduced data sets with feature selection. The decrease of the size of the vector space gives the J48 classifier a positive effect (7.33% performance increase between the cases in Table 1 and Table 3).

Naïve Bayes classifiers (CNB, MNB and NB) show the best performance in all different cases. The performances of these classifiers decrease when bigram and trigram representations are used. The use of stemming and feature reduction on data set increased the classification accuracy of MNB and NB classifiers (MNB 2.16% and NB 7.5%).

If bigram and trigram representations are used, the performance of classifiers decreases considerably. The one reason for this can be the sparse data that is caused by the limited size of corpus. If the number of N-gram words increases in the feature space of a data collection sufficiently, its probability estimate will be better.

As a future work we are planning to work on larger data sets with more number of classes. We will also use semantic web technologies to represent our documents as a set of concepts rather than as a set of independent words. Additionally, we also plan to use interpolation techniques [17] that enable to work with different N-Gram models (unigram, bigram, trigram representations) simultaneously to get better classification results.

10. References

[1] Amasyalı M.F. and Yıldırım T., "Otomatik Haber Metinleri Sınıflandırma", SIU, Kuşadası, 2004.

[2] Amasyalı M.F. and Diri B., "Automatic Turkish Text Categorization in Terms of Author, Genre and Gender", Springer 11th International Conference on Applications of Natural Language to Information Systems-NLDB2006, Austria, LNCS, 3999, 2006.

[3] Yıldız H.K., Gençtav M., Usta N., Diri B. and Amasyalı M.F., "A New Feature Extraction Method for Text Classification", IEEE 15th Signal Processing and Communication Applications Conference, SIU, Eskişehir, Türkiye, 2007.

[4] Güven Ö., Bozkurt Ö. and Kalıpsız O., "Advanced Information Extraction with n-gram based LSI.", Proceeding of World Academy of Science, Engineering and Technology, Vol. 17, Dec. 2006.

[5] Özgür L., Güngör T. and Gürgen F., "Adaptive Anti-Spam Filtering for Agglutinative Languages: A Special Case for Turkish", *Pattern Recognition Letters*, 25(16), pp.1819-1831, 2004.

[6] Sebastiani F., "Machine learning in automated text categorization", *ACM Comput. Surv.,* 34(1), pp.1-47, Mar. 2002.

[7] Witten I.H. and Frank E., *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[8] John G.H. and Langley P., "Estimating continuous distributions in Bayesian classifiers.", Proc. 11th Conf. Uncertainty in Artificial Intelligence, pp. 338-345, 1995.

[9] Rennie J., Lawrence S., Teevan J. and Karger D., "Tackling the Poor Assumptions of Naive Bayes Text classifiers.", In Proceedings of ICML, 2003.

[10] McCallum A. and Nigam K., "A comparison of Event Models for Naive Bayes Text Classification.", In AAAI'98, workshop on learning for text categorization pp. 41–48, 1998.

[11] Quinlan R., "C4.5: Programs for Machine Learning.", Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[12] Aha D.W., Kibler D. and Albert M.K., "Instance-based Learning Algorithms.", *Machine Learning*, 6 , pp. 37-66, 1991.

[13] Han J. and Kamber M., *Data Mining: Concepts and Techniques*, 2nd Edition, The Morgan Kaufmann Series in Data Management Systems, 2005.

[14] Oflazer K., "Two-level Description of Turkish Morphology.", Literary and Linguistic Computing, 9(2), pp.137-148, 1994.

[15] Hankemer J., "Lexical Representation and Process.", The MIT Press, Chapter Morphological Parsing and the Lexicon, 1989.

[16] http://www.sil.org/pckimmo/ntnlp94.html

[17] Jurafsky D. and Martin J.H., "N-Grams", *Speech and Language Processing*, 2nd Edition, Pearson Education, 2009.