Turkish Keyphrase Extraction Using KEA

Nagehan Pala
Dept. of Computer Engineering
Bilkent University
06800 Bilkent Ankara, TURKEY
Email: nagehan@cs.bilkent.edu.tr

İlyas Çiçekli
Dept. of Computer Engineering
Bilkent University
06800 Bilkent Ankara, TURKEY
Email: ilyas@cs.bilkent.edu.tr

***Abstract*—Keyphrases provide semantic metadata to summarize and characterize documents. Unfortunately, there are many digital documents especially on the Internet that do not have a list of assigned keyphrases. Assigning keyphrases to these documents manually is a tedious process and requires knowledge of the subject. *Automatic Keyphrase Extraction* solves this problem. In this paper, we present implementation of *Keyphrase Extraction Algorithm (KEA)* for Turkish as well as extending it with new features to improve its performance.**

I. Introduction

A keyword, from a very simple point of view, a word that occurs with a high frequency within a text document. If a word appears with a higher frequency in a document, we can say this word provides semantic metadata to summarize and characterize the document. Keywords often contain two or more contiguous words so we prefer to call them *keyphrases*.

Keyphrases can be used for different purposes.

- They can be used for *summarization*. The reader quickly understands what the document is about and whether it is in his fields of interest.
- They can be used for *indexing*. The reader quickly finds relevant documents.
- They can be used for *more precise searches*. The query term can be searched in the keyphrases list of the documents rather than searching them in the full text of the documents. This yields better results.

Although many academic articles have a list of keyphrases that are assigned by their authors, unfortunately, there are also many digital documents especially on the Internet that do not have such a list. Assigning keyphrases to these documents manually is a tedious process and requires knowledge of the subject. *Automatic keyphrase extraction* solves the problem. Automatically extracting the important phrases from the document and automatically generating the important phrases from the document are not the same. Automatically generated keyphrases may or may not be in the document although automatically extracted keyphrases should appear in the document. Our approach that is defined in the remainder of the paper is *automatic keyphrase extraction*.

We approach the problem as a *machine learning* task. Machine learning is a broad subfield of *artificial intelligence* and is concerned with the development of algorithms and techniques that allow computers to *learn*. The algorithms can be categorized in several groups, based on the desired outcome of the algorithm. One of them is *supervised learning* that generates a function from training data and this function maps inputs to desired outputs. This is the classical machine learning problem of *learning from data*. Training data consist of both the input vectors and their outputs. Supervised learning algorithms first takes the training data and creates a function. After generating the function, it takes the test data. Test data consist of only input vectors. It predicts the desired output of the given input by using this function.

We can use a supervised learning algorithm to extract keyphrases from a document. A document contains a set of phrases that must be classified either as a keyphrase or not. First, we prepare a training data which contain the documents with the assigned keyphrases. After creating the function, the algorithm takes the test data which contain the documents without the assigned keyphrases. We expect the algorithm to extract keyphrases from these documents. *Naïve Bayes* is a supervised machine learning algorithm. Our approach is based on Naïve Bayes algorithm which is explained in detail in the following section.

Keyphrase Extraction Algorithm(KEA) [1] is used for automatically extracting keyphrases from documents. KEA is developed by a group from Waikato University. Their concern is extracting reasonable summaries from text documents. KEA is developed for English documents. We use it to extract keyphrases from Turkish text documents by making some changes, for example changing its *stemmer* and *stopwords list*. The differences between the original KEA algorithm and our algorithm are explained in detail in Section II.

The experiments in this paper are based on a document collection which consists of 60 Turkish articles [3]. The corpora is explained in Section III.

Our performance measure is the number of correctly identified author-assigned keyphrases. After extracting the keyphrases from the test data, we calculate the total correctly identified keyphrases with respect to the total author assigned keyphrases. Performance analysis of Turkish keyphrase extraction is given in Section IV.

We conclude (in Section V) that KEA algorithm can be used to extract keyphrases from Turkish text documents. It can provide useful metadata where none existed before.

II. KEA: Keyphrase Extraction Algorithm

KEA algorithm uses a supervised learning algorithm, so it has two stages:

1) Training Stage: KEA takes the training data and creates a model which is used to extract the keyphrases. Training data consist of the documents with the author-assigned keyphrases.
2) Extraction Stage: KEA takes the data (or test data) which consist of the documents without the author-assigned keyphrases and extracts the keyphrases of these documents by using the created model.

These two stages have common steps to perform their task. These steps are:

1) Select candidate phrases.
   a) Clean input.
   b) Identify the phrases.
   c) Case-fold and stem the phrases.
2) Calculate these features for each selected candidate phrase.
   a) Calculate *TF×IDF* feature.
   b) Calculate *First Occurrence* feature.
   c) Calculate *Relative Length* feature.[1]

First, these common steps are described in detail and then the two stages are explained. KEA is developed to extract keyphrases from English text documents. We change some parts of KEA algorithm to make it suitable for automatic keyphrase extraction from Turkish text documents. The differences are identified in the related steps.

A. Selecting Candidate Phrases

Candidate phrases are selected by applying three steps. First, the input text is cleaned. Second, candidate phrases are identified. Finally, identified phrases are stemmed and case-folded.

1) Input Cleaning: Input files are filtered by applying these rules:

- Punctuation marks, brackets, and numbers are changed with phrase boundaries.
- Apostrophes are removed.
- Hyphenated words are split in two.
- The tokens that do not contain letters are removed.
- Acronyms are handled as a single token.

After applying these rules to an input document, we get the sequence of words which consists of at least one letter.

2) Phrase Identification: Candidate phrases are identified by applying the following rules:

- Candidate phrases are limited to a certain maximum length.
- Candidate phrases cannot be proper names.[2]
- Candidate phrases cannot begin or end with a stopword.

[1]This feature is not calculated in the original KEA algorithm.

[2]It is true for original KEA algorithm, but it may or may not be applied in our implementation.

One of the differences between KEA that is used for English documents and our implementation is the stopword list. We add a stopword list which contains 114 words [4]. It can be expanded, actually we used a simple list.

For example, *yapı, su, işleri, müdürlüğü, yapı ve su, yapı ve su işleri, su işleri müdürlüğü* phrases are identified from *yapı ve su işleri müdürlüğü* (if certain maximum length is three words). None of them begins or ends with *ve* because it is a stopword.

3) Case-Folding and Stemming: Case-folding and stemming is the last step. First, all words are case-folded. This results in word matching effectively being case-insensitive. Its major drawback is in handling of acronyms (e.g. "TİK"(Türkiye İstatistik Kurumu) & "tik"). Then, all words are stemmed. Another difference between KEA that is used for English documents and our implementation is the stemming algorithm. We use *Zemberek* [5] to stem the words.

An important difference between English and Turkish is that Turkish is a highly agglutinative language [6]. This means that English does not tend to stack more than four or five affixes, whereas Turkish can have words with nine or ten affixes. There are two types of suffixes: inflectional suffixes and derivational suffixes. Discarding the inflectional suffixes is simple. However, a word can have more than one derivational suffix. When such a word is stemmed, which one should be selected? For example, *gözlükçü* has two derivational suffixes. We select the minimum length word. In this case, it is *göz*.

Stemming is also used for comparing the results. We compare the stemmed version of KEA extracted keyphrases and the stemmed version of author-assigned keyphrases. If stemmed versions are the same, then we say that they match. For example, "gözler" and "gözümün" match.

Stemmed and unstemmed versions of a phrase are saved together. If the phrase is selected as a keyphrase then the unstemmed version is presented to the user.

B. Feature Calculation

Two features are calculated for each candidate phrase by the original KEA algorithm. They are *TF×IDF* and *first occurence*. Our implementation uses one more feature with these features.

1) TFxIDF: This feature is calculated by using two frequencies: the frequency of a phrase in a document and the frequency of that phrase in general use. If a phrase is used in a document with a high frequency, the probability of being a keyphrase of this phrase increases. If a phrase is used in general use with a high frequency, the probability of being a keyphrase of this phrase decreases.

The *TF×IDF* for phrase P in document D is:

$$\text{TF} \times \text{IDF} = \frac{\text{freq}(P, D)}{\text{size}(D)} \times -\log_2 \frac{\text{df}(P)}{N} \quad . \tag{1}$$

Components of (1) are as follows:

1) freq(P,D) is the number of times P occurs in D
2) size(D) is the number of words in D

3) df(P) is the number of documents containing P in the global corpus
4) N is the size of the global corpus

By using (1), *TF×IDF* feature is claculated for each candidate phrase.

2) First Occurrence: First occurrence feature is calculated as the number of words that precede the phrase's first appearance is divided by the total number of words in the document.

3) Relative Length: Relative length feature is calculated as the number of characters in the phrase is divided by the number of characters in the candidate phrase that has the maximum. Original KEA algorithm does not calculate and consider this feature. We add this feature for performance analysis. It is also used by Turney [2].

C. Training Stage

In this stage, KEA takes training data which contain the text documents and their author assigned keyphrases. After selecting candidate phrases and calculating their feature values, KEA marks these phrases as keyphrases or non-keyphrases by using the author assigned keyphrases. If a phrase is given in the author assigned keyphrases list, then this phrase is marked as a keyphrase, otherwise it is marked as a non-keyphrase. Being a keyphrase or not being a keyphrase is the class value for *Naïve Bayes* algorithm. It generates a model using training data to predict the class.

D. Extraction Stage

In this stage, KEA takes data (or test data) which contain only text documents. After selecting candidate phrases and calculating their feature values, KEA determines the probability of being a keyphrase for each candidate phrase by using the generated model. *Naïve Bayes* algorithm calculates (2) and (3) for each candidate phrase.

$$P[\text{yes}] = \frac{Y}{Y+N} \times P_{\text{TFxIDF}}[t|\text{yes}] \times P_{\text{distance}}[d|\text{yes}] \times P_{\text{relLength}}[r|\text{yes}] \quad (2)$$

$$P[\text{no}] = \frac{N}{Y+N} \times P_{\text{TFxIDF}}[t|\text{no}] \times P_{\text{distance}}[d|\text{no}] \times P_{\text{relLength}}[r|\text{no}] \quad (3)$$

Components of the equation are as follows:

1) *t* is *TF×IDF*
2) *d* is *distance*
3) *r* is *relative length*
4) *Y* is the number of positive phrases in the training documents
5) *N* is the number of negative instances in the training documents

After calculating these equations, their results are substituted in (4).

$$p = \frac{P[\text{yes}]}{P[\text{yes}] + P[\text{no}]} \quad . \quad (4)$$

Candidate phrases are ranked according to the values calculated from (4). If the probabilities of two candidate phrases are equal, their *TF×IDF* values are compared to rank them. Finally, the phrases that are the subparts of other phrases whose rank is higher are deleted.

III. Corpora

The experiments in this paper are based on a document collection which consists of 60 Turkish articles. The articles are obtained from the online archives of *Journal of The Faculty of Engineering and Arhitecture of Gazi University* [3]. First, we convert them from PDF format to text format. Then, we made the following changes on these articles:

- Abstract and keyphrases parts written in English are deleted from the articles.
- Keyphrases parts written in Turkish are moved to *.key* documents whose names are the same as the article.

The average size of the documents is approximately 3450 words.

IV. Experiment

This section presents four experiments with Turkish keyphrase extractor. In the first experiment, we assess the effect of relative length feature. In the second experiment, we vary the number of keyphrases to be output. In the third experiment, the effect of document length is assessed. Finally, overall effectiveness comparison between Turkish and English keyphrase extraction is shown in the last experiment.

A. Experiment 1: Effect of Relative Length Feature

In this experiment, we analyze whether *relative length* feature is useful for Turkish keyphrase extractor or not. For this purpose, the results of two different configurations are compared. In the first configuration, *relative length* feature is not calculated. In the second configuration, this feature is calculated. We used 50 text documents for training and 10 text documents for testing. The same set of documents are used for training and testing in these two configurations.

The results are given in Table I. The first column shows the number of keyphrases that are assigned by the author. The second column shows the number of keyphrases that are assigned by KEA. The third column of the table shows the number of common keyphrases that are assigned by the author and the first configuration of KEA. The fourth column shows the number of common keyphrases that are assigned by the author and the second configuration of KEA. Each row shows the results for one document. The last row shows the total numbers.

The performance of two configurations are calculated as the number of total common keyphrases is divided by the total number of author assigned keyphrases. The performance of the first configuration is $\frac{3}{39}$ which is %7. The performance of the second configuration is $\frac{11}{39}$ which is %28. The performance of the second configuration is higher than the performance of the first configuration, so *relative length* feature is useful for Turkish keyphrase extraction.

TABLE I
EXPERIMENT 1 RESULTS

Author assigned	Extracted	Match TR	Match TR-RL
4	5	1	1
4	5	0	2
6	5	0	1
3	5	0	1
3	5	0	0
4	5	1	1
4	5	0	2
3	5	0	0
4	5	0	2
4	5	1	1
39	**50**	**3**	**11**

TABLE II
EXPERIMENT 2 RESULTS

Author	Common 5	Common 10	Common 15	Common 20
4	1	2	3	3
4	2	2	2	2
6	1	2	2	2
3	1	1	1	1
3	0	1	1	2
4	1	3	4	4
4	2	3	3	3
3	0	0	1	1
4	2	2	2	2
4	1	2	2	2
39	**11**	**18**	**21**	**22**

B. Experiment 2: Effect of Number of Extracted Keyphrases

We carried out a series of tests to determine how the number of phrases output affects performance. We used 50 text documents for training and 10 text documents for testing. We calculated the performance measures for 5, 10, 15 and 20 phrases. The results are given in Table II.

For example, look at the document which is given in the 6th row. The phrases that are extracted by the KEA for 15 phrases output and the phrases that are assigned by the author is given in Table III. The phrases that are matched with the author assigned keyphrases are italicized. The rank of *paslanmaz çeliklerin* is 3, so it can be found by KEA for 5 phrases output. The rank of *kesme kuvvetleri* is 6 and the rank of *yüzey pürüzlülük* is 8, so they can be found by KEA for 10 phrases output. The rank of *kesici takımların kaplanması* is 11, so it can be found by KEA for 15 phrases output. As there are no more keyphrases that can be found, KEA for 20 phrases output is unnecessary.

C. Experiment 3: Effect of Document Length

In this experiment, we extract keyphrases from the abstracts of the documents. The performance comparison between full text and abstract based keyphrase extraction is given in Table IV.

The average values were obtained from six-fold cross validation. We partitioned the data into six equal-sized subsets. Each subset was used five times for training and only one time for testing. To obtain average value, effectiveness values of each run were summed and the result was divided by six.

TABLE III
MATCHED KEYPHRASES

Author assigned keyphrases	KEA assigned keyphrases
kesme kuvvetleri	kesici takım
kesici takım kaplaması	paslanmaz
paslanmaz çelik	*paslanmaz çeliklerin*
yüzey pürüzlülüğü	östenitik paslanmaz çeliklerin
	kesme hızının
	kesme kuvvetleri
	pürüzlülük
	yüzey pürüzlülük
	paslanmaz çeliğin işlenmesinde
	kaplanmış kesici
	Kesici takımların kaplanması
	kesme hızlarında kesici
	sementit karbür
	kaplanmış kesici takımla
	talaş

TABLE IV
EFFECT OF DOCUMENT LENGTH

Extracted	Avg. Match Full Text	Avg. Match Abstract
5	1.05	1.02
10	1.42	1.35
15	1.63	1.45
20	1.75	1.48

TABLE V
OVERALL EFFECTIVENESS

Extracted	Avg. Match EN	Avg. Match TR	Avg. Match TR-RL
5	0.93	0.40	1.05
10	1.39	0.50	1.42
15	1.68	0.57	1.63
20	1.88	0.62	1.75

The abstract based keyphrase extraction performs not as good as full text based keyphrase extraction. The reason is that entire document contains more author assigned keyphrases than the document abstract does.

D. Experiment 4: Overall Effectiveness

This experiment shows the overall effectiveness. We give the effectiveness results for Turkish and English [1] in Table V. Columns of the table represent the following items respectively.

1) Number of keyphrases extracted
2) Average matches with author keyphrases for English
3) Average matches with author keyphrases for Turkish without Relative Length feature
4) Average matches with author keyphrases for Turkish with Relative Length feature

The average values were also obtained via six-fold cross validation.

When we use relative length feature, the average matches for Turkish and English keyphrase extraction are similar.

V. CONCLUSION

The performance of KEA for English is stated in [1]:

> Our results show that KEA can on average match between one and two of the five keyphrases chosen by the author.

Actually, if *relative length* feature is not calculated, the results are worse than this result. However, if we factor *relative length* feature in, the results are almost the same.

REFERENCES

[1] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin and C.G. Nevill-Manning, "KEA: Practical automatic keyphrase extraction", *Proceedings of the Fourth ACM Conference on Digital Libraries*, pp. 254-256, 1999.

[2] P. D. Turney: "Learning to extract keyphrases from text", *Technical Report ERB-1057, National Research Council, Institute for Information Technology*, 1999.

[3] *Journal of The Faculty of Engineering and Architecture of Gazi University*, Vol. 21 Nr. 1, Nr. 2, Nr. 3, Nr.4 and Vol. 20 Nr. 1, Nr.2, Nr. 3, 2006.

[4] (2007, May) [Online] http://www.ranks.nl/stopwords/turkish.html

[5] (2007, May) [Online] https://zemberek.dev.java.net/

[6] K. Oflazer, "Two-level description of Turkish morphology", *Literary and Linguistic Computing*, 9(2), 1994.