Turkish Keyphrase Extraction Using Multi-Criterion Ranking

Bahadır Özdemir
Dept. of Computer Engineering
Bilkent University
06800 Bilkent Ankara, Turkey
Email: bozdemir@cs.bilkent.edu.tr

Ilyas Cicekli
Dept. of Computer Engineering
Bilkent University
06800 Bilkent Ankara, Turkey
Email: ilyas@cs.bilkent.edu.tr

***Abstract*—Keyphrases have been extensively used for indexing and searching in databases and information retrieval systems. In addition, they provide useful information about semantic content of a document. In this paper, we propose an algorithm for automating Turkish keyphrase extraction. Several features of candidate phrases are exploited and form the extraction task as a problem of finding optimal set of candidate phrases. We use multi-criterion ranking to tackle this problem.**

I. Introduction

Finding a relevant document in library databases or on Internet is a challenging task that requires successful clustering and indexing. Keyphrases are extensively used for indexing, categorization and summarizing of documents. Searching a word in the keyphrase list of a document instead of full content reduces search time. In addition, they provide semantic data about content of the document. Unfortunately, the great majority of documents do not have author assigned keyphrases [1]. Automating keyphrase extraction provides benefits for manual assignment of keyphrases by indexers.

The remaining of this paper is structured as follows. In section II, It starts with a short description of what is done previously. Our keyphrase extraction algorithm is explained in section III. The results of experiments are given in section IV. Finally, a short discussion about our method and comparison with previous methods are provided in section V.

II. Related Work

There are several methods for automating keyphrase extraction in English. On the other hand, only two methods have been developed for Turkish keyphrase extraction. They are Turkish keyphrase extraction with KEA [2] and Turkish keyphrase extractor (TurKeyX) [3]. Both of them are extended from English keyphrase extraction algorithms.

In [1], KEA (Keyphrase Extraction Algorithm) has two stages to extract keyphrases: Training stage and extraction stage. In the first stage, a model to identify keyphrases is created using training documents where author assigned keyphrases are known. In extraction stage, keyphrases of a new document are chosen according to this model. The original KEA uses two features that are calculated for each candidate phrase: $TF \times IDF$ (Term Frequency $\times$ Inverse Document Frequency) and the distance from the beginning of the document to the first occurrence of the candidate phrase. KEA uses Naïve Bayes technique to learn two sets of numeric weights from the discretized feature values. The model created in training stage determines the overall probability that each candidate is a keyphrase. Finally, candidate phrases are ranked according to this probability value and the ones with higher probability are returned as keyphrases of the document. In [2], KEA is adopted for Turkish keyphrase extraction. The difference with the original KEA is using Turkish stemmer and stopword list instead of English counterparts. Furthermore, a new feature is added to the algorithm. The new feature is relative length, calculated as the number of characters in the phrase divided by the length of the longest candidate phrase.

Another keyphrase extraction method called B&C is proposed by Barker and Cornacchia in [4]. B&C uses noun phrase heads to extract document keyphrases. B&C exploits statistics of noun phrases, noun phrase heads and noun phrase lengths. In contrast with KEA, it does not need a training corpus. B&C selects top N head nouns according to their frequencies. For each head noun, all noun phrases that contain same word as its head are collected. Each noun phrase is assigned a score calculated as the product of its frequency and length. K high scored phrases are selected as the keyphrases of the document. In [3], the approach of TurKeyX to extract Turkish keyphrases is quite similar to B&C's. The difference is that TurKeyX method uses statistics of noun phrases and noun phrase heads. Moreover, TurKeyX also borrows some features computed in KEA and Turney's GenEx [5].

Our method is more similar to TurKeyX than KEA in the sense that it does not require training data so the performance does not depend on the corpora used. In addition, most of the features in our method were also used in TurKeyX. However, the ranking strategies of two methods in extraction phase are different.

III. Extracting Keyphrases using Multi-Criterion Ranking

Our keyphrase extraction technique consists of two stages. In the first stage, candidate phrases are extracted from documents and features are calculated for each candidate phrase. In the second stage, a Hasse diagram is created from the set of candidate phrases. Finally, an optimal set of keyphrases is selected using the ranking method described in [6].

A. Selecting Candidate Phrases

We select candidate phrases in a similar way used in [3]. We consider every noun phrase in the document as a candidate phrase. To identify noun phrases, we employ a supervised Turkish part of speech tagger (POS) [7]. The tagger returns most possible POS tag for each token (word, punctuation etc) in the document. The tagger also provides morphological analysis of each word. We use these analyses for stemming. Next step is to find noun phrases using POS tags. Noun phrases consist of one head noun and zero or more premodifying adjectives or nouns [4]. A sub-phrase of a noun phrase can also be a noun phrase. For this reason, each sub-phrase is considered as a different candidate. As mentioned in [3], this method is not an ideal NP-Chunker. Many phrases identified by this method are not eligible for being a keyphrase. Therefore, several filters are used to reduce the number of candidate phrases.

First filter is about stopwords. We use a list of 112 stopwords [8]. It is also used in KEA and TurKeyX. If a phrase starts or ends with a stopword, it will be removed from the list of candidates. Another filter limits the number of words in a candidate phrase. In our implementation, we eliminate phrases which consist of more than five words. In addition, we remove phrases that includes a word having only one or two characters and not recognized by both morphological analyzer in [7] and Zemberek [9]. Last two filters are based on n-gram approach and require stem-based frequency analysis of phrases. A noun phrase can be a combination of noun phrases like

$$\alpha = \beta_1 \cdots \beta_k, \text{ for } k \geq 2$$

where α and β_i are noun phrases for $1 \leq i \leq k$. As expected, α and its sub-phrase, β_i, should be in the candidate list. Our aim is to determine most suitable candidates from such a combination. Our first frequency filter eliminates sub-phrases of a suitable candidate. The frequency of a phrase is equal or less than the frequency of its sub-phrase. However, the ratio of phrase frequency to sub-phrase frequency should be high if the phrase is more suitable candidate than its sub-phrase. For every combination α, we remove its sub-phrase β_i from the list if

$$\frac{f(\alpha)}{f(\beta_i)} > T_{high}, \text{ for } 1 \leq i \leq k \tag{1}$$

where $f(\cdot)$ denotes stem-based frequency of its argument.

Another case is that a suitable candidate can occur next to another noun phrase in a document. In such a condition, the ratio of frequencies should be low. For every combination α, we remove the phrase α from the list if

$$\exists i, \frac{f(\alpha)}{f(\beta_i)} < T_{low}, \text{ for } 1 \leq i \leq k. \tag{2}$$

In our implementation, the values of T_{high} and T_{low} are determined empirically as 0.5 and 0.1, respectively.

B. Feature Calculation

The features used in our algorithm are well-known features used in various keyphrase extraction algorithms. Six features are calculated for each candidate phrase. They are dispersion, first occurrence, head noun first occurrence, phrase length, frequency and head noun frequency. The first one is borrowed from [10], and the others are similar to the features in [3]. In computation of all features, comparison of words is case-insensitive and stem-based.

1) Dispersion: In [11], a measure is developed for the condensation of the term over textual units. Content-bearing terms have a tendency to be clustered or clumped. If a term is clustered, fewer textual units would contain the term. The measure is based on the differences in probability of finding a content-bearing phrase and not content-bearing phrase in a textual unit. The expected number of documents containing the phrase can be found as

$$E = D \times \left[1 - \left(1 - \frac{1}{D}\right)^T\right] \tag{3}$$

where D is the number of documents in the corpus, and T is the total number of occurrences of the phrase. Condensation clustering of the phrase is defined as

$$M_c = \frac{N}{E} \tag{4}$$

where N is number of documents that actually contain the phrase. The dispersion would be one if the phrase is randomly distributed over documents. The dispersion would be less than one if it is clustered.

2) First Occurrence: First occurrence feature is the number of words that precede the phrases first appearance.

3) Head Noun First Occurrence: Head noun first occurrence feature is the number of words that precede the first appearance of the phrases head noun.

4) Phrase Length: Phrase length is the number of words in the phrase.

5) Frequency: The frequency of the phrase in the document.

6) Head Noun Frequency: The frequency of the head noun of the phrase in the document.

C. Multi-Criterion Ranking

Ranking with Hasse Diagrams is a known method in statistics [6]. However, it has not been used in keyphrase extraction yet. General approach for ranking objects is to combine all features into one measure and sort them with respect to this measure. This method is used in [3]. The problem of combining measures into one measure is to determine weights of measures. Another approach for ranking objects is to use partially ordered sets (posets) described in [6].

1) Partially Ordered Sets and Hasse Diagram: The candidate phrases will be denoted by $a, b, c, \ldots$ etc. The aim is to make comparative statements about two given candidate phrases a and a' according to their feature values $(I_1, I_2, \ldots, I_6)$ and $(I'_1, I'_2, \ldots, I'_6)$ respectively. If $I'_j \geq I_j$

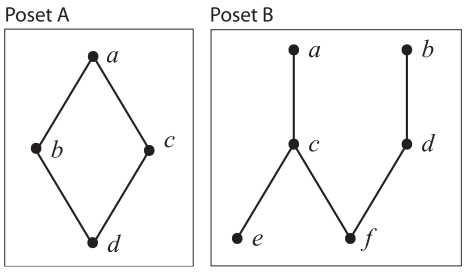

Fig. 1. Hasse diagrams for two different posets.

for all j, then a' is intrinsically "better" or "bigger" than a, and we write $a' \geq a$. In the case of a poset, we write $a < b$ if $a \leq b$ but $a \neq b$. One further relation required for ranking in a poset is covering relation. Candidate phrase b covers candidate phrase a if $a < b$ and there is no candidate phrase x for which $a < x < b$. We write $a \prec b$ when b covers a.

Hasse Diagram is a way of presenting posets. A point (or vertex) is plotted for each candidate phrase. Candidate phrase b is located higher on the paper than a whenever $a < b$. The vertices a and b are connected by a straight line segment (an edge) whenever $a \prec b$. Hasse diagrams for two different posets can be seen in Figure 1. Covering relations in Poset B are $c \prec a$, $e \prec c$, $f \prec c$, $d \prec b$ and $f \prec d$.

If we try to sort the vertices in Poset A in Figure 1, possible ranks of vertex a is only 1 and possible ranks of vertex d is only 4. On the other hand, possible ranks of vertices b and c are 2 and 3. Thus, there are two possible rankings: a, b, c, d and a, c, b, d. In the poset literature, rankings are called linear extensions of a poset. We can use the probability of possible ranks in order to sort a poset. Rank-interval of a phrase can be computed using its upper and lower sets. Given S as a set of elements, then the upper set of element $a \in S$ is defined as

$$U_a = \{x \in S : x > a\} \tag{5}$$

Similarly, the lower set is defined as

$$L_a = \{x \in S : x < a\} \tag{6}$$

The rank interval of element a can be defined as

$$|U_a| + 1 \leq r \leq |S| - |L_a| \tag{7}$$

where there is a ranking that assigns rank r to element a. The collection of all linear extensions of S is denoted as Ω. The set Ω is finite but generally very large. Members of Ω are denoted by the symbol ω, and the rank which ω assigns to $a \in S$ is written as $\omega(a)$. The rank-frequency distribution of element a is given by

$$f_a(r) = \#\{\omega \in \Omega : \omega(a) = r\} \tag{8}$$

and corresponding cumulative rank-frequency (CRF) distribution becomes

$$\begin{aligned} F_a(r) &= f_a(1) + f_a(2) + \cdots + f_a(r) \\ &= \#\{\omega \in \Omega : \omega(a) \leq r\}. \end{aligned} \tag{9}$$

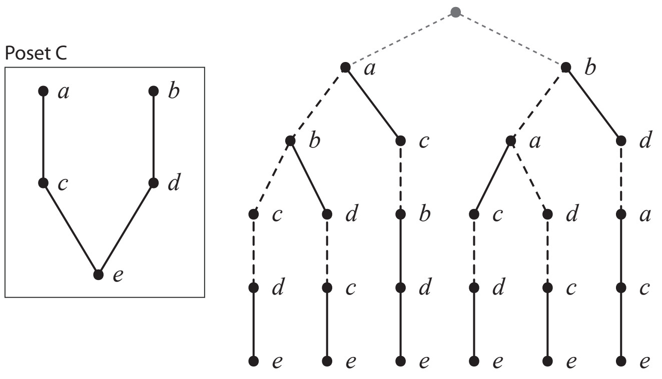

Fig. 2. Hasse diagram for Poset C and its linear extension decision tree.

Poset C and its linear extensions decision tree are given in Figure 2. There are six linear extensions of Poset C and three of them assign rank 1 to element a, notationally $f_a(1) = 3$. Similarly, $f_a(2) = 2$ and $f_a(3) = 1$.

Patil and Taillie propose CRF operator for linearizing a poset [6]. The operator uses cumulative rank-frequency distributions as new indicator values and creates a new poset from the original one. This operation is applied iteratively until the poset becomes linear. In other words, the final poset has only one linear extension that gives the ranking of objects.

The number of linear extensions of a poset increases with factorial complexity when the number of objects in the poset increases. As a result, enumerating all linear extensions of bigger posets becomes computationally impossible [6]. For bigger posets, Patil and Taillie propose Markov Chain Monte Carlo (MCMC) sampling [6].

2) Implementation Details: For each indicator value, higher values should imply higher possibility to be a keyphrase. For this reason, we negate the values of dispersion, first occurrence, head noun first occurrence and phrase length. Furthermore, there exists an exceptional case for phrase length. Phrases consisting of less number of words are generally more preferred as keyphrases by authors. However, two-word phrases are more preferred than one-word phrases [12]. For this exceptional case, phrase length of two-word phrases is considered as one and phrase length of one-word phrases is considered as two in the implementation.

In experiments of our keyphrase extraction algorithm, posets of candidate phrases are usually big posets. Therefore, we first reduce the number of candidate phrases by selecting at most fifty candidate phrases according to the midpoints of their rank-intervals. Afterwards, we apply MCMC sampling technique in linearization of the posets. Finally, first ten phrases in the sorted list of the poset are offered to user as keyphrases of the document.

IV. Experiments

A. Corpora

Two different corpora are used for experiments. First one is a collection of Turkish scientific papers obtained from the online archives of Journal of The Faculty of Engineering and

TABLE I
SAMPLE KEYPHRASE EXTRACTION FOR GAZI UNIV. JOURNAL CORPUS

Author Assigned	KE-MCR
İmalat Hücresi	*Uzman Sistem*
Uzman Sistemler	*İmalat Hücresi*
Esnek İmalat Sistemleri	Robot
Yapay Zeka	Tezgah
	İşlem Sıralaması
	Uzman Sistemin
	İmalat Sistemi
	Makinadan
	Algoritmalar
	Boşaltma

TABLE II
PERFORMANCE RESULT FOR GAZI UNIV. JOURNAL CORPUS

	Number of Extracted Phrases	Average Number of Matches	Avg. Number of Author Assigned Keyphrases
KE-MCR	5	1.03	4.0
	10	1.40	
TurKeyX	5	0.90	4.0
	10	1.37	
KEA-TR	5	1.05	3.9
	10	1.42	

Architecture of Gazi University. The corpus was created by Pala [2]. The corpus consists of 60 papers in text format and their author-assigned keyphrases. Second corpus is a collection of news articles taken from the web pages of newspapers and news portals. There are totally 30 news articles. This corpus was created by Kalaycilar [3].

B. Performance Evaluation

Our algorithm returns 10 phrases as keyphrases of the document (Table I and Table III). Phrases written in italic match with author assigned keyphrases, or they are considered as same. Partially matched phrases are not treated as correct match except that they refer to same person; such as *Kerimov* for *Islam Kerimov*. In addition, acronym of a keyphrase is also considered as correct match. Sample results of keyphrase extraction using multi-criterion ranking (KE-MCR) for Gazi University Journal corpus can be seen in Table I.

Performance comparison of KE-MCR with TurKeyX [3] and Turkish Keyphrase Extraction with KEA (KEA-TR) [2] for Gazi University Journal articles is given in Table II. Performance of algorithms is given for the first 5 and 10 phrases extracted by the algorithm. All documents in Gazi University Journal corpus are used in testing for KE-MCR and TurKeyX algorithms. On the other hand, 50 documents are used in training and only 10 documents are used in testing for KEA-TR algorithm.

Similar comparison is made for news corpus. Sample keyphrase extraction for news corpus is given in Table III and performance result is shown in Table IV. All documents in the news corpus are used in testing for both algorithms.

TABLE III
SAMPLE KEYPHRASE EXTRACTION FOR NEWS CORPUS

Author Assigned	KE-MCR
Tony Blair	*Tony Blair*
Katolik Kilisesi	Katolik
Eski İngiltere Başbakanı	*Katolik Kilisesine*
	Bayan Blair
	Yılında
	Kilise Sistemi
	Blair
	Başbakan
	İngiltere
	İşçi Partisi

TABLE IV
PERFORMANCE RESULT FOR NEWS CORPUS

	Number of Extracted Phrases	Average Number of Matches	Avg. Number of Author Assigned Keyphrases
KE-MCR	5	1.03	3.3
	10	1.47	
TurKeyX	5	0.97	3.3

V. CONCLUSION AND FUTURE WORK

We propose a method for automating keyphrase extraction using multi-criterion ranking. As seen in Table II and Table IV, our method outperforms TurKeyX in both corpora. In addition, the performance of our method is very close to KEA-TR. However, the performance of KEA-TR method depends on training corpus. On the other hand, our method does not require training corpus, so the performance does not change drastically from one corpus to another. For example, the performance of our method in both corpora is equal when five phrases are extracted and the results are very close for extracted ten phrases.

We developed a method for Turkish keyphrase extraction. However, the method is suitable for extending to languages other than Turkish. Using counterparts for morphological analyzer, part of speech tagger and stopword list is sufficient for extension.

We expect that using a reliable NP-Chunker increases performance considerably. Furthermore, we think that adding semantic features like node degree in KEA++ [12] may enhance the performance.

ACKNOWLEDGMENTS

This work is partially supported by The Scientific and Technical Council of Turkey Grant "TUBITAK EEEAG-107E151".

REFERENCES

[1] I. Witten, G. Paynter, E. Frank, C. Gutwin, and C. Nevill-manning, "Kea: Practical automatic keyphrase extraction," in *Proceedings of Digital Libraries 99 (DL'99)*, Feb 1999, pp. 254–255.

[2] N. Pala and I. Cicekli, “Turkish keyphrase extraction using kea,” in *Proceedings of the 22nd International Symposium on Computer and Information Sciences (ISCIS 2007)*, Ankara, Turkey, Nov 2007, pp. 1–5.

[3] F. Kalaycilar and I. Cicekli, “Turkeyx: Turkish keyphrase extractor,” in *Proceedings of the 23rd International Symposium on Computer and Information Sciences (ISCIS 2008)*, Istanbul, Turkey, Oct 2008, pp. 1–4.

[4] K. Barker and N. Cornacchia, “Using noun phrase heads to extract document keyphrases,” in *Advances in Artificial Intelligence: 13th Biennial Conference of the Canadian Society for Computational Studies of Intelligence, AI 2000*, Quebec, Canada, May 2000.

[5] P. Turney, “Learning algorithms for keyphrase extraction,” *Information Retrieval*, vol. 2, pp. 303–336, 2000.

[6] G. Patil and C. Taillie, “Multiple indicators, partially ordered sets, and linear extensions: Multi-criterion ranking and prioritization,” *Environmental and Ecological Statistics*, vol. 11, no. 2, pp. 199–228, June 2004.

[7] T. Daybelge and I. Cicekli, “A rule-based morphological disambiguator for turkish,” in *Proceedings of Recent Advances in Natural Language Processing (RANLP 2007)*, Borovets, Bulgaria, 2007, pp. 145–149.

[8] (2009, April) Turkish stopwords. [Online]. Available: http://www.ranks.nl/stopwords/turkish.html

[9] (2009, April) Zemberek. [Online]. Available: https://zemberek.dev.java.net

[10] J.-L. Wu and A. M. Agogino, “Automating keyphrase extraction with multi-objective genetic algorithms,” in *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS’04) - Track 4*. Washington, DC, USA: IEEE Computer Society, 2004, p. 40104.3.

[11] A. Bookstein, S. Klein, and T. Raita, “Clumping properties of content-bearing words,” *Journal of the American Society for Information Science*, vol. 49, pp. 49–2, 1998.

[12] O. Medelyan and I. Witten, “Thesaurus based automatic keyphrase indexing,” in *JCDL ’06: Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, 2006, pp. 296–297.