Available online at www.sciencedirect.com

ScienceDirect

Information Processing and Management 43 (2007) 1705–1714

INFORMATION PROCESSING & MANAGEMENT

www.elsevier.com/locate/infoproman

Using lexical chains for keyword extraction

Gonenc Ercan, Ilyas Cicekli [*]

Department of Computer Engineering, Bilkent University, Bilkent 06800, Ankara, Turkey

Received 3 July 2006; received in revised form 9 January 2007; accepted 10 January 2007
Available online 2 March 2007

Abstract

Keywords can be considered as condensed versions of documents and short forms of their summaries. In this paper, the problem of automatic extraction of keywords from documents is treated as a supervised learning task. A lexical chain holds a set of semantically related words of a text and it can be said that a lexical chain represents the semantic content of a portion of the text. Although lexical chains have been extensively used in text summarization, their usage for keyword extraction problem has not been fully investigated. In this paper, a keyword extraction technique that uses lexical chains is described, and encouraging results are obtained.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Keyword extraction; Lexical chains; Natural language processing; Machine learning

1. Introduction

Keywords can be considered as brief summaries of a text. Therefore it is possible to think of them as a set of phrases semantically covering most of the text. Although a summary of a text is capable of providing more information about the text than keywords of the text, the summary may not be suitable for some applications due to the complex structure of sentences. Keywords are not replacements for summarization but alternative summary representations that could be consumed by other applications more easily. Since they are concise representations of the underlying text, it is possible to use them in different applications such as indexing in search engines or text categorization.

Keywords enable readers to decide whether a document is relevant for them or not. They can also be used as low cost measures of similarity between documents. Unfortunately, a great portion of existing documents available today does not have keywords available for them. Considering the fact that it is a hard and time consuming task to assign keywords to documents, it is desirable to automate this task by machine learning and natural language processing (NLP) techniques.

Authors can assign keyphrases for their documents, and those keyphrases might or might not occur in the text. In automatic keyphrase extraction, most indicative phrases in a document are selected as keyphrases for

[*] Corresponding author. Tel.: +90 312 2901589; fax: +90 312 2664047.
E-mail addresses: ercangu@cs.bilkent.edu.tr (G. Ercan), ilyas@cs.bilkent.edu.tr (I. Cicekli).

0306-4573/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ipm.2007.01.015

that document. Thus, automatic keyphrase extraction algorithms are limited with phrases that appear in the text. More general form of keyphrase extraction is keyphrase generation which does not select phrases from the document, but generates and assigns keyphrases for the document. In this paper, we concentrate on "keywords" instead of "keyphrases" to emphasize the fact that keyphrases can be composed of more than one word, and we only extract keywords.

We believe that a keyword of a text should be semantically related with the words of the text. A lexical chain for a text contains a subset of the words (word senses) in the text. The words in the lexical chain are semantically related. A lexical chain may cover a small or big portion of the text. The number of words and the number of semantic relations among the words can be different for each lexical chain. The coverage and size of a lexical chain can indicate how well the lexical chain represents the semantic content of the text. So, we believe that a keyword which represents the semantic content of the text should be selected from the words of a lexical chain which represents the most of the semantic content of the text. In this paper, we present a keyword extraction method such that it uses the features based on lexical chains in the selection of keywords for a text.

Keyword extraction is highly related to automated text summarization. In text summarization, most indicative sentences are extracted to represent the text. In keyword extraction, most indicative keywords are extracted to represent the text. In both of these problems, the features like word frequencies, cue phrases, position in text, lexical chains and discourse structure are exploited to discover a pattern representing importance in a text. In this paper, we aim to explore the effect of lexical chains in keyword extraction, when the problem is treated as a supervised machine learning task. This learning task uses features based on the lexical chains of words. Since we can build lexical chains for words only (not for phrases) using the WordNet ontology (Fellbaum, 1998), we concentrate on the keyword extraction problem instead of keyphrase extraction.

Although we have experimented with different classifiers such as Naive Bayes, we obtained better results with the decision tree induction algorithm C4.5 (Quinlan, 1993). For this reason, we have used C4.5 in order to represent the keyword extraction problem as a learning task. We used C4.5 with two different sets of features. In our baseline system, we used only the text features (without using any feature based on the lexical chains of words). In the second case, C4.5 was used with the features based on the lexical chains in addition to the features used in the baseline system. Then we compare the results of these two versions. We have obtained better results when the features based on the lexical chains were used.

We first present the related work on keyword extraction and lexical chains in Section 2. Then lexical chains and creation of lexical chains are described in Section 3. After lexical chain based features that are used in our keyword extraction system are explained in Section 4, we discuss the results of our keyword extraction method in Section 5. Finally, we give some concluding remarks in Section 6.

2. Related work

A summarization system tries to identify significant information that is important enough to be in the summary. In order to identify important topics and ideas in the document, summarization systems try to use clues in the document. These clues can be sentence level features such as the position and length of the sentence, and the word level features such as word frequency and position of words. The summarization methods that use supervised learning techniques identify the sentences containing important information using the clues as the indication of the importance, and these clues are learned from the training data. For example, Kupiec, Pedersen, and Chen (1995) and Teufel and Moens (1997) use sentence level features to classify important sentences. Kupiec also uses word frequency information for the classification task. They have not used any feature based on semantics of words. In our system, we use semantic word features based on lexical chains of words to determine important keywords in the document.

Treating keyword extraction as a supervised learning algorithm has been also investigated in (Frank, Paytner, Witten, Gutwin, & Nevill-Manning, 1999; Fellbaum, 1998; Hulth, 2004; Turney, 2000). Turney applied the decision tree algorithm C4.5 and a genetic algorithm (GenEx) for keyword extraction (Turney, 2000). Turney basically uses two features for these algorithms: word position in text and word frequency. Turney experimentally has shown that using bagging decision trees in C4.5 algorithm improves the performance of C4.5 for keyword extraction. The fitness function of the genetic algorithm GenEx is the accuracy of extractor, and the population consists of the parameter values. However GenEx is computationally expensive in training time. Turney experimentally shows that GenEx generalizes well to different domains, removing the necessity to train for different domains.

Kea (Frank et al., 1999) is another algorithm, which treats keyword extraction as a supervised learning task. Kea uses TFxIDF (term frequency * inverse document frequency) and normalized word position as features. TFxIDF is the word frequency in the document normalized by the domain frequency, and it is a domain specific feature. Kea has obtained competing results with GenEx on general training. Kea is much faster than GenEx and performs better when it is trained with domain dependent data. Kea uses Naïve Bayesian algorithm to learn to classify phrases. Although Kea does not use a parser to detect phrases, all possible word sequences up to three words are treated as possible phrases. Some restrictions are applied to filter some of word sequences. For example, punctuation symbols are not considered as words, and phrases cannot start with stop words. In our system, we only extract keywords and all nouns in the document are treated as possible keywords.

Since keyword extraction and automated text summarization tasks are highly related tasks, some techniques used in text summarization algorithms can be used in keyword extraction algorithms. Barzilay and Elhadad (1997) have shown that the features based on *lexical chains* are good features for text summarization. Lexical chains can be used to analyze lexical cohesion (Morris & Hirst, 1991) in a text. Every text has some level of lexical cohesion, as lexical cohesion is made up of semantic relationships between words. Lexical chains are built using the semantic relations between words. Lexical chains are also used for text summarization by other researchers (Angheluta et al., 2002).

Silber and McCoy (2000) describe an efficient method for building lexical chains. This work encourages the use of lexical chains in different applications, since it provides an efficient method for building lexical chains which is an exponential problem. Lexical chains have been used in different NLP problems such as word sense disambiguation, text summarization, text segmentation and topic tracing.

We also represent the keyword extraction problem as a supervised learning task, and use the decision tree algorithm C4.5 as a classifier in our keyword extraction system. Although the lexical chain based clues are used in text summarization, we are the first to suggest the lexical chains in keyword extraction. Since the lexical chains of words represent the lexical cohesion in the text, they give important clues about the semantic content of the text. In this paper, we demonstrate that the lexical chains can play an important role in the selection of keywords that semantically represent the text. We use Silber's approach when the lexical chains are built for the given text.

3. Building lexical chains

Lexical cohesion (Morris & Hirst, 1991) occurs among sequences of related words, and the *lexical chains* of a text can capture the lexical cohesive structure of the text. Lexical chains are identified by using relationships between word senses. In order to build lexical chains, word senses and semantic relations between words should be known. WordNet (Fellbaum, 1998) is a database providing such knowledge, and we use WordNet in our lexical chain building algorithm.

We use synonym sets, hyponym/hypernym and meronym trees of WordNet in order to find relations between two word senses in the creation of lexical chains. Building lexical chains is an exhaustive method. This is mainly due to the fact that a word can have many senses. Correct sense of the word must be chosen. In other words, we should disambiguate word senses.

In Barzilay and Elhadad (1997) and Barzilay (1997), an exhaustive algorithm is described to disambiguate word senses while building the lexical chains. In a cohesive and meaningful text, the word sense that is related with more word senses should be the correct sense. Barzilay propose building all possible lexical chains for the text and choosing the highest scoring interpretation of the text. However, this is hard to compute as the number of interpretations may increase exponentially with each word. On the other hand, using a greedy approach could lead to misinterpretations. Barzilay uses pruning of interpretations as the number of interpretations increases, but still this is a computationally expensive process. Another technique used in (Barzilay, 1997) is the usage of text segments (windows of 7 sentences) in order to reduce the complexity of the algorithm.

Silber and McCoy (2002) describe another efficient method that evaluates lexical chains by word sense disambiguation. Silber builds metachains for a text, by finding the relationships between words, but not building the lexical chains. For each word, the word sense that has more than one relationship with other words is selected and the other senses of the word are removed. After disambiguating all the words in the text, the lexical chain is built. In both of Barzilay's and Silber's algorithms, lexical chains in two different segments are merged if there is a word re-iteration or a synonym relation.

If there is only one lexical chain for a text, this means that all the words in the text are related. In general, a text can be associated with more than one lexical chain. In this case, the words of the text are grouped according to relatedness into subsets where each subset is indicated by a lexical chain.

We have used the approach described in (Silber & McCoy, 2002) with minor changes. In (Silber & McCoy, 2002), the WordNet database is re-indexed to access it more efficiently. We have not done this re-indexing, instead we created flat relationship lists for each word (up to 3 levels of depth). This technique enabled us to check for relations in linear time and enabled us to find a relation between two word-senses faster. Before building lexical chains we have to identify nouns in the text. We have used only the relations between nouns, since they provide more information about the subject, and keywords usually appear in text as nouns. The part of speech (POS) tagger, that is used in order to identify nouns in the text, is Maxent Tagger (Toutanova & Manning, 2000; Toutanova, Klein, Manning, & Singer, 2003).

The WordNet relations used in our version of lexical chain builder are synonym, hypernym/hyponym and meronym. The algorithm in (Silber & McCoy, 2002) uses the same relations except meronym, and they also consider hypernym siblings in the construction of the lexical chains. The system in (Barzilay & Elhadad, 1997) uses exactly the same features that we use.

Although we use lexical chains in order to extract keywords of the documents, the lexical chains can also be used in word sense disambiguation (WSD). We did not measure the performance of our lexical chain builder for the word sense disambiguation task. However, since our lexical chain builder is similar to Silber's approach (Silber & McCoy, 2002) and WordNet features that we use are exactly same as Barzilay's features (Barzilay & Elhadad, 1997), we expect that the accuracy of our system will be similar to the results of their systems. Galley and McKeown (2003) report that Silber's algorithm has 53% accuracy for WSD problem, and Barzilay's algorithm has 57% accuracy.

English is a productive language for compound nouns. Barzilay identifies noun compounds in the text and queries WordNet for the phrase (Barzilay & Elhadad, 1997). If the word does not appear in WordNet it uses the head noun for the identified phrase. Silber did not handle noun compounds (Silber & McCoy, 2002). WordNet has limited noun compounds, and keyphrases are usually domain dependent phrases that are not present in WordNet. So, compound nouns are not used in our lexical chains, and only individual words appear in our lexical chains. For this reason, we have only concentrated in the extraction of keywords instead of extraction of keyphrases.

Each node in a lexical chain is a word sense of a word, and each link can be synonym/reiteration, hyponym/hypernym, or meronym relation between two word senses. For example, the simple lexical chain in Fig. 1 represents relations among the selected word senses of the words appearing in that lexical chain. In Fig. 1, the solid lines are hyponym relations, and the dotted lines are synonym relations.

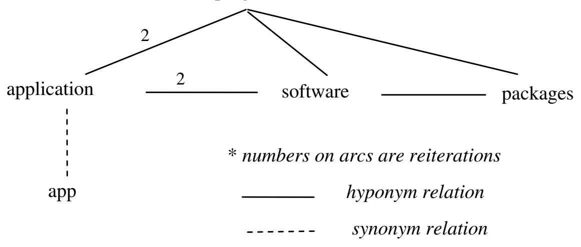

Fig. 1. A simple lexical chain.

4. Feature extraction

Each word (noun) in a text is associated with different features. We use these features with C4.5 decision tree induction algorithm. The features that are used in our baseline system are the first occurrence of the word in text (first occurrence position), the number of word occurrences in text (frequency), and the last occurrence of the word (last occurrence position). The first two features are also used in (Turney, 2000). Although Turney uses some other features (Turney, 2000), we prefer only these features since most of other features are related with keyphrases and we only deal with keywords. Extracting these three features is straightforward. The last occurrence feature and the first occurrence feature represent the lifespan of words in the text. These three features are strictly coupled with the symbolic representation of the word. When C4.5 is used with these three features, it makes our baseline system. In order to see the effects of the features based on the lexical chains, the results of the baseline system is compared with the system with the lexical chain features.

In addition to these three features in our baseline system, we have experimented with different features that can be derived from lexical chains in the text. After experimenting with different features based on the lexical chains, we have decided to use the following four features based on lexical chains in our main system.

Lexical chain score of a word: A word can be a member of more than one lexical chain as it can appear in the same text with different senses. We choose the most representative lexical chain for this word. In other words, after scoring each lexical chain of the word, we choose the score of the one with the maximum score as *the lexical chain score of the word*. The *score of a lexical chain* depends on the relations appearing in the lexical chain. For each relation between word senses, weights are assigned intuitively. These assigned weights for the relations can be seen in Table 1.

This feature is calculated by scoring all the relations of the lexical chain, whether it is directly related with the word we are dealing with or not. For example, if the lexical chain in Fig. 1 is the lexical chain with the maximum score for the word ‘‘program’’, the score of that lexical chain will be used as the lexical chain score of the word ‘‘program’’. The score of this lexical chain is equal to 59 ($=7 \times 7 + 10$) since there are seven hyponym relations and one synonym relation (each reiteration is counted as a separate relation).

Direct Lexical chain score of a word: This feature is calculated by scoring only the relations that belong to the word. In other words, we only use the arcs directly connected to the word sense. *Direct lexical chain score* for the word ‘‘program’’ in the lexical chain is equal to 28 ($=4 \times 7$) since there are only four hyponym arcs directly connect to the word ‘‘program’’ in Fig. 1.

Lexical chain span score of a word: The *span score of a lexical chain* depends on the portion of the text that is covered by the lexical chain. The covered portion of the text is considered to be the distance between the first occurrence position of a lexical chain member (word) which occurred first in the text and the last occurrence position of a lexical chain member (word) which occurred last in the text. The span score of a lexical chain is computed by finding the difference between these two positions. For example, in order to evaluate the span score of the lexical chain in Fig. 1, we consider all positions of all the words in the lexical chain. If *pfirst* is the first position and *plast* is the last position among those positions, the span score of the lexical chain is equal to *plast* - *pfirst*. The *lexical chain span score of a word* is the span score of the lexical chain with the maximum score, containing the word.

Direct lexical chain span score of a word: In order to evaluate the *direct lexical chain span score of a word*, we evaluate the direct span scores of all lexical chains containing the word for that word. The score of the lexical chain with maximum score will be the *direct lexical chain span score of the word.* The *direct span score of a lexical chain for a word* is computed same as the lexical chain span score except that we consider only the words that are directly related with the word in the lexical chain. For example, let us assume that the lexical

Table 1
Weights of relations in lexical chains

Relation	Description	Weight
Synonym/reiteration	Same meaning	10
Hypernym/hyponym	Generalization/specialization	7
Meronym	Member-of/has-a/part-of	4

chain in Fig. 1 is the lexical chain with the maximum direct span score for the word "program". In order to evaluate the direct span score of the lexical chain (for the word "program") in Fig. 1, we consider only the portion of the text covered by the words *program*, *application*, *software* and *packages* (not *app*) which are directly related with the word "program".

The reader may note that these features can only be extracted if the word is in WordNet. If the word does not appear in WordNet, these features are left as missing attributes. It is hard to extract these features for compound nouns because WordNet does not contain most of the compound nouns.

5. Learning to extract keywords

We have experimented C4.5 decision tree induction algorithm with different features to learn to extract keywords in the documents. Our learning algorithm is trained to find keywords that appear in keyphrases with their correct forms. For instance, if "reaction time" is an author-assigned keyphrase, our learning algorithm is trained to extract both "reaction" and "time" as keywords. This differs from the work in (Frank et al., 1999; Turney, 2000) since they train to extract exactly "reaction time" in their algorithms. In testing phase, our algorithm marks a found keyword as correct if the keyword appears as an author-assigned keyword or a word of an author-assigned keyphrase.

There are variations of C4.5 in the literature. Experimentally, we have seen that bagging with decision trees, C4.5 provides better results for keyword extraction problem. For this reason, our version of C4.5 also uses bagging with decision trees. Bagging is the process of classifying the instances with multiple classifiers. In bagging technique, an instance is classified with multiple classifiers, and the average classification probability is used to classify the instance. Bagging decreases the variance, increasing the accuracy. Same as our observations, Turney (2000) also reports that bagging improves the performance of the decision tree algorithm for keyphrase extraction.

In order to evaluate the effect of lexical chain features used in keyword extraction, we use two different sets of features. In our baseline system, we have extracted keywords by using only the following three features which are not based on lexical chains.

First occurrence position – the first occurrence position of a word in the text.
Word frequency – the frequency of a word (how many times a word appears).
Last occurrence position – the last occurrence position of a word in the text.

Then we compared these results with the results obtained by adding lexical chain features. We used the following four lexical chain features which are described in the previous section.

lexical chain score of a word,
direct lexical chain score of a word,
lexical chain span score of a word,
direct lexical chain span score of a word.

Thus, our main learning algorithm uses seven features. We experiment our learning algorithm with different subsets of these seven features, and the results are given in this section.

One of our corpora is the same corpus used in (Frank et al., 1999; Turney, 2000). This corpus consists of 75 journal articles, and we have used 50 of these journal articles for training and 25 for testing. The texts in the corpus are full texts (not abstracts). We have extracted 1, 5, 10, 15 keywords for each document. We have observed that all the lexical chain features described above improve precision results.

Table 2 shows the experiment results that we have obtained. The first row of the table gives the results when no lexical chain features are used, and the second row gives the results when all seven features including four lexical chain features are used. We have observed that the precision of the classifier improves significantly with lexical chain features (lexical chain, direct lexical chain, lexical chain span and direct lexical chain span). For example, we only get 17% precision when we extract 5 keywords if we do not use lexical chain features. On the other hand, we get 45% precision if we use lexical chain features. This means that almost half of found

Table 2
Precision results – effects of lexical chain features with the corpus of full length texts

Number of keywords per document	1	5	10	15
Precision for baseline system (with three features) (%)	9	17	18	14
Precision for the system with all seven features (%)	64	45	30	26

keywords appear in author-assigned keyphrases when 5 keywords are extracted. These results indicate that the lexical chain features improve the precision significantly.

Although the same corpus is also used in (Turney, 2000), we cannot directly compare Turney's results in (Turney, 2000) with the results presented in Table 2. The system in (Turney, 2000) is similar to our baseline system, but they use more features than our baseline system, and they also extract keyphrases (not keywords). They got 28% precision when they extract 5 keyphrases. This is better than our baseline system's result (17% precision), but worse than the result of our main system (45% precision). Of course, these can be misleading because of the reasons given above.

Table 3 shows the precision results for two different subsets of lexical chain features. Although all of the lexical chain features improve precision results, the span features improve more than other lexical chain features according to the results presented in Table 3.

We have also experimented with some other lexical chain based features in addition to the four lexical chain based features that described in this paper. We observed that these four lexical chain based features together with the features in the baseline system are the best feature combination among the features that we have experimented for keyword extraction. Although there can be other lexical chain based features that may also reflect the lexical cohesion of the text, we did not get better precision results from the other lexical chain based features that we have experimented (Ercan, 2006).

According to the results in Tables 2 and 3, our precision values drop when the number of extracted keywords increases. There are two main reasons for this precision drop. First, the average number of keywords in author assigned keyphrases is 5.8 keywords per document in the corpus, and some of author assigned keywords do not appear in the document. When the number of extracted keywords is more than 5, the precision drops because the set of extracted keywords will definitely contain keywords that are not author assigned keywords. Second, the extracted keywords are selected depending on the probabilities that are assigned by the classifier. The first selected keyword has the highest probability, and the probabilities drop when we move down in the list. When the number of extracted keywords increases, the keywords with lower probabilities are also included in the result set, and this increases the number of misses. Thus, these two reasons can lead a precision drop in our experiments.

The other corpus that we have experimented with, is the corpus used in (Witten, Paynter, Frank, Gutwin, & Nevill-Manning, 1999). This corpus consists of 155 abstracts (not full texts). We trained with 110 and tested with 45 of these abstracts. The results that we obtained are given in Table 4. Although lexical chains still seem to provide better results, its effect is not as high as in the full-text experiment. According to the results in Table 4, the precision is only improved 4% (from 16% to 20%) in the extraction of five keywords in the corpus of abstracts. On the other hand, the precision is improved 28% (from 17% to 45%) in the extraction of five keywords in the corpus of full length texts as given in Table 2. This means that the lexical chain based features significantly improve the performance in the extraction of the keywords from full length texts while the performance improvement is less significant in the extraction of keywords from abstracts (short length texts). The reason for this observation can be that the lexical cohesion clues are lower in short length texts, and the lexical

Table 3
Precision results – using only some of lexical chain features with the corpus of full length texts

Number of keywords per document	1	5	10	15
Precision – only lexical chain and direct lexical chain features (%)	27	28	22	17
Precision – only lexical chain span and direct lexical chain span features	64	30	22	20

Table 4
Precision results – effects of lexical chain features with the corpus of abstracts

Number of keywords per document	1	5	10	15
Precision for baseline system (with three features) (%)	20	16	13	14
Precision for the system with all seven features (%)	27	20	17	16

chains are weaker since the semantically related words are not widely used in short length texts. In full length texts, the same concept is emphasized with using more semantically related words in the text. If there are a lot of semantically related words in a text, the lexical chains of the text will be stronger because there will be more relations between word senses. Thus, the lexical chain based features play more important role in the selection of keywords.

A sample document's abstract is given in Table 5. The keyphrases assigned by the author are given in Table 6. The five keywords extracted by our algorithm from the full text of that abstract are given in Table 7, and the correct classifications are given in boldface. The extracted keywords are sorted in descending order, and the five keywords with highest scores are given in Table 7. Although the word ‘‘Parkinson’’ is not in the correct

Table 5
The abstract of a sample document

ABSTRACT – Redundancy of the motor control system gives the central control structures options for solving everyday motor problems. The choice of particular control patterns is based on priorities (coordinative rules) that are presently unknown. Motor patterns observed in unimpaired young adults reflect these priorities. We hypothesize that in certain atypical conditions, which may include disorders in perception of the environment and decision-making, structural or biochemical changes within the central nervous system, and/or structural changes of the effectors, the central nervous system may reconsider its priorities. A new set of priorities will reflect the current state of the system and may lead to different patterns of voluntary movement. In such conditions, changed motor patterns should be considered not pathological but rather adaptive to a primary disorder and may even be viewed as optimal for a given state of the system of movement production. Therapeutic approaches should not be directed towards restoring the motor patterns to as close to ‘‘normal’’ as possible but rather towards resolving the original underlying problem. We illustrate this approach with movements in amputees and patients with Parkinson's disease, dystonia, and down syndrome.

Table 6
The author keywords of the sample document

Voluntary movement
Motor control
Movement disorders
Coordination
Posture
Pre-programming
Parkinson's disease
Down syndrome

Table 7
The extracted keywords of the sample document (the correct keywords are in boldface)

Motor	**Syndrome**
Rule	**Parkinson**
Abstract	**Motor**
Control	**Movement**
Problem	**Control**
Baseline system with three features	The system with all seven features

form, it is marked as a correct result. This is the best example for our lexical features. In general, our main system with seven features gets 45% precision for full texts and 20% for abstracts when 5 keywords are extracted.

6. Conclusion and future work

This paper describes a keyword extraction method to investigate the benefits of using lexical chain features in keyword extraction. According to the results that are obtained, the lexical chain features improve the precision significantly in the keyword extraction process. Although lexical chains have been used in different application domains, to the best of our knowledge we are the first to use lexical chains in the keyword extraction problem.

In this paper, we have tried to extract keywords only since WordNet does not contain too many word phrases. We are working on how to extract keyphrases in addition to keywords. In order to able to do this, we are investigating ways of putting phrases into lexical chains. In fact, some phrases are already present in WordNet. For the phrases that are not in WordNet, a possible approach is to use the head noun of the phrase to represent that phrase and put the head noun in the lexical chains. In fact, Barzilay (1997) uses a similar technique when building lexical chains. Barzilay uses lexical chain based features to select sentences. Since we are going to use them to select phrases, our representation of the lexical chains must be more accurate. We are also considering to use other knowledge sources such as WikiPedia to relate phrases in the lexical chains.

Acknowledgement

This work is partially supported by The Scientific and Technical Council of Turkey Grant "TUBITAK EEEAG-105E065".

References

Angheluta, R., De Busser, R., & Moens M.-F. (2002). The use of topic segmentation for automatic summarization. In U. Hahn, D. Harman (Eds.), *Proceedings of the workshop on automatic summarization 2002, Philadelphia, PA, USA* (pp. 66–70).

Barzilay, R. & Elhadad, M. (1997). Using lexical chains for text summarization. In *Proceedings of the ACL'97/EACL'97 workshop on intelligent scalable text summarization, Madrid, Spain*.

Barzilay, R. (1997). Lexical chains for summarization. Masters Thesis, Ben Gurion University of Negev.

Ercan, G. (2006). Automated text summarization and keyphrase extraction. Masters Thesis, Bilkent University.

Fellbaum, C. (Ed.). (1998). *WordNet: an electronic lexical database*. MIT Press.

Frank, E., Paytner, E., Witten, I. H., Gutwin, C., & Nevill-Manning, C. G. (1999). Domain specific keyphrase extraction. In *Proceedings of the sixteenth international joint conference on artificial intelligence* (pp. 667–668). Morgan Kaufmann.

Galley, M., & McKeown, K. (2003). Improving word sense disambiguation in lexical chaining. In *Proceedings of IJCAI*, 486–1488.

Hulth, A. (2004). Enhancing Linguistically Oriented Automatic Keyword Extraction. In *Proceedings of the human language technology conference/North American chapter of the Association for Computational Linguistics Annual Meeting (HLT/NAACL 2004), Boston, May 2004*.

Kupiec, J., Pedersen, J. O., & Chen, F. (1995). A trainable document summarizer. In *Proceedings of SIGIR'95* (pp. 68–73). ACM Press.

Morris, J., & Hirst, G. (1991). Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics, 17*(1), 21–48.

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kauffmann.

Silber, H. G., & McCoy, K. F. (2000). Efficient text summarization using lexical chains. In *Proceedings of the 5th international conference on intelligent user interfaces, New Orleans, Louisiana* (pp. 252–255).

Silber, H. G., & McCoy, K. F. (2002). Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics, 28*(4), 487–496.

Teufel, S., & Moens, M. (1997). Sentence extraction as a classification task. In Inderjeet Mani & Mark T. Maybury (Eds.), *Proceedings of ACL/EACL97-WS*. Spain: Madrid.

Toutanova, K., & Manning, C. D. (2000). Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the joint SIGDAT conference on empirical methods in natural language processing and very large Corpora (EMNLP/VLC-2000), Hong Kong* (pp. 63–70).

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL* (pp. 173–180). Canada: Edmenton.

Turney, P. (2000). Learning to extract keyphrases from text. *Journal of Information Retrieval, 2*(4), 303–336.

Witten, I. H., Paynter, G. W., Frank, E., Gutwin, C., & Nevill-Manning, C. G. (1999). KEA: Practical automatic keyphrase extraction. In *Proceedings of DL '99*, 12, 254–256.