An Efficient WordNet-Based Summarizer for Large Text Documents

Raymond K. Wong Chit Sia
School of Computer Science & Engineering
University of New South Wales
Sydney, NSW 2052, Australia
wong@cse.unsw.edu.au

Technical Report
UNSW-CSE-TR-0311
June 2003

SCHOOL OF COMPUTER SCIENCE & ENGINEERING
THE UNIVERSITY OF NEW SOUTH WALES

Abstract

The current information overload problem has called for the need to develop an automatic text summarization system. This paper presents an efficient sentence-based extraction summarizer which can be used for the above purpose. Lexical chains were used as a basis and knowledge resources such as WordNet and a sentence boundary disambiguation tool were integrated to the system for better performance.

Three different summary extraction heuristics were used and compared. An intrinsic evaluation which involved the comparison of our summarizer with a commercial product to the human written abstracts was performed. The results obtained have been encouraging, and it is found that our system favors the human judgement than the other system.

The algorithm used in this paper demonstrated a linear runtime behavior. This not only suggests a positive position in the scalability of our system but also its potentiality in handling documents of longer length.

1 Introduction

Throughout the last decade, the growing popularity of Internet has stimulated the exponential growth of document-based information online. The wide availability of Internet and the rapidly expanding global information network has also meant that information retrieval is now a much more familiar and convenient process to casual users. However, this explosion of information and interest of information-seekers has resulted in a well-recognized information overload problem. There is already no time to read everything, and yet it is not unusual that during searching information on the Internet, a large proportion of the results are often not relevant to the query. This leads to the consequence that the user ends up spending a significant amount of time reading through details of documents that are only "potentially" relevant. This problem has called for attention of the need to research and development of systems to automatically summarize text and since then, it has become the focus of considerable interest and investment in both the research and commercial sectors. By building automatic text summarization systems, it is hoped that they can act as a decision or navigation tool to help the users in deciding which articles are most suitable for their needs in a more effective and efficient way.

In this paper, a sentence-based extraction summarization system which takes in a single document from any domain was developed to produce an indicative summary as output. The approach taken was initially similar to that taken by Barzilay and Elhadad [1]. It is based on the use of lexical chains, which can be computed without requiring "deep" text understanding while can still be able to determine the context of the text at the same time. However, different to Barzilay and Elhadad, except the semantic relatedness of words in the text was considered during the summarization process, the existence of text title and the actual distance between words in the text were also considered as factors in determining the "aboutness" of the input document. Furthermore, different heuristics for detecting relations between words were developed to deliver a more efficient system, which has shown to increase its feasibility in handling longer documents.

2 Related Work

Automated text summarization is not a new idea. Earliest systems were developed in the late 1950s, using mainly the word-frequency-based approach. Since then, several different approaches to text summarization have been implemented and aimed to improve the quality of the summarized text. From their level of processing, different approaches can be characterized into three categories: surface-level, discourse-level and entity-level.

2.1 Surface-Level

Surface-level approaches tend to represent information in terms of shallow features which are then selectively combined together to yield a salience function used to extract information. [25] These features include word frequency statistics, location, background, cue words and phrases [5, 11, 19, 23, 34, 42].

2.2 Discourse-Level

The second category is the discourse-level summarization systems. These systems attempt to model the global structure of the text, and its relation to communicate goals [25]. Clearly the discourse-based summarization technique [7, 26, 27] falls into this category.

2.3 Entity-Level

The last category of summarization approach is the entity-level. Entity-level approaches tend to build an internal representation of the source text, modelling the text entities and their relationships. They aim to use the connectivity in the text to determine the salient part of the text [25]. Examples of connectivity include word co-occurrence, proximity of text units, co-reference, syntactic relations, similarity and thesaural relationships among words. Information-extraction-based and Cohesion-based summarization systems are two systems that are under this category:

2.3.1 Information-extraction-based

Information-extraction-based systems [28, 29] are knowledge-based systems that apply knowledge of the domain to characterize specific conceptual knowledge of a text. They are usually used to generate abstracts that concern very specific aspects, such as when, who, what, why, etc, of some events. One drawback is that they are usually not domain independent.

2.3.2 Cohesion-based

The assumption made by the cohesion-based summarization systems is that important words, sentences, and paragraphs are the highest connected entities in elaborate graph-like representations of text. Words in the text are assigned weights based on their “statistical and formal-semantical characteristics” while the word-word relations are detected on the basis of lexical similarity and by using a thesaurus (Paice [35]). According to this heuristic, the most important sentences in the source text are those that are related to the largest number of other sentences. Salton and Singhal [39] used term repetition to construct a weighted graph with nodes being the paragraphs and the weighted arcs representing the similarity between them.

Another class of this cohesion-based system is the use of lexical chain, whose concept was first introduced by Morris and Hirst [31]. Barzilay and Elhadad [1] explored the use of lexical chains and used WordNet as their knowledge base. They claimed that by using lexical chains, the readers will be able to get a better indication of the topic of the text compared to simply picking the most frequent words in the text (i.e. frequency-based). In their system, chains are built in two processes: the source text is first segmented using TextTiling method proposed by Hearst [9], then for each segment, nouns and noun phrases in the text found by a shallow parser are grouped into lexical chains based on their cohesion relationships with each other using WordNet. Then, chains between segments are merged if they contain a word in the same sense in common. Strong chains are chosen based on the number and weight of different relations in the chain and a fixed length summary is then generated based on these chains.

3 System Overview

The automatic text summarization system developed in this paper can be divided into four stages: preprocessing, lexical chains computing, strong chains extractor and finally, the sentence extractor.

Input

Input to the summarization system is a single document from any domain. The title of the document can also be attached, but this is not compulsory. The idea of allowing the inclusion of title file is based on the assumption that, a good title should have positive influence on the context of the document. On the other hand, when there is no title present, or when the title is too general or unrelated, the judgement in determining the “aboutness” of the document should not be affected.

Preprocessing

The first step of the summarization process is text segmentation. The role of the text segmenter is to divide the input document into segments that address the same topic. Under the assumption that most articles have good division to segments marked as paragraphs, the division used in this paper is based on paragraphs. Regular expressions were used to detect paragraphs boundaries.

Since this summarization system is a sentence-based system, each sentence in the document has to be disambiguated. In order to detect sentence boundary at a higher accuracy, Reynar and Ratnaparkhi's sentence boundary detector [37] was employed in this paper. Below gives a detailed description of the issue of sentence boundary disambiguation and tool developed by Reynar and Ranaparkhi.

Sentence Boundary Disambiguation

In order to improve the accuracy in detecting sentence boundaries, a tool developed by Reynar and Ratnaparkhi was used in this paper. The system is pre-trained on 39441 sentences of Wall Street Journal text from the second release of the Penn Treebank and requires no hand-crafted rules, lexica, part of speech tags or domain specific information.

Lexical Chains Computing

Before lexical chains can be constructed, candidate words must first be chosen. Candidate words are defined as words that have the potential to contribute to the capturing of the context of the document. Each candidate word is exploded into senses, using information obtained from a thesaurus. The lexical knowledge database used in this paper was WordNet. Lexical chains are then built based on the semantic relatedness among these senses (e.g. synonyms, antonyms, hypernyms/hyponyms, meronyms/holonyms, etc). Since a word can have several senses, in building lexical chains, the sense of each word in the document will be able to be disambiguated. Again, WordNet was used as the knowledge source for this word sense disambiguation process.

Strong Chains Extractor

Once the lexical chaining process is completed, strong chains must be chosen to capture the most important concept of the input document. Strong chains are determined by a chain-score scaling system developed in this paper. Each chain is initially associated with a chain-score, based on the degree of semantic relatedness among words in the chains, what this score scaling system does is that, it takes into account the word-to-word distance as well. Each chain-score is scaled according to this factor. In doing so, the distance between word instances in the input document is allowed to affect the word's contribution to a chain as well, aiming to develop a higher quality system.

Sentence Extractor

The final step of the summarization system is to extract sentences from the input document and form a summary as output. There are many ways one can produce a summary based on the strong chains extracted. In this paper, three different heuristics have been proposed for determining which sentences should be included in the summary. Out of the three, two are flexible, which means that all sentences in the document are ranked according to its importance, where the degree of importance depends on the heuristic used. Once the rank of each sentence is available, users are allowed to choose the length of summary wanted, based on the number of sentences they want to be included in the summary. On the other hand, the remaining heuristic gives a fixed length summary. This length is calculated according to the number of strong words (i.e. representative words) found in the strong chains and is about 5-15% of the original document on average.

4 Lexical Chains Building

Lexical chains are chains of semantically related words. They are computable without requiring "deep" text understanding, and exist in non-domain specific text. They provide a constrained easy-to-determine representation of context for interpreting words, concepts, and sentences. Hence, it makes sense to use it as clues to the structure of the text, and it forms the basis of the summarization system developed in this paper.

We built the lexical chains by following the following steps:

1. Check if there is a title file. If there is, insert the title to the beginning of the text
2. Select all the candidate words from the text by calling ***selectNouns***
3. For each segment, call the ***segmentChainBuilder(segment candidate word list)*** to get a list of chains for that segemnt
4. Merge all the chains obtained from each segment.

5 Strong Chains Extraction

Once the lexical chains can be computed, the function ***chainScaling*** can be called to scale the chain-score according to the distance of the candidate words in the chain and return a list of strong chains (where score is above a certain threshold) for the text.

The score scaling system allows the distance between word instances in the source text to affect the word's contribution to a chain. The scaling system is as shown in Table 1. The scaling system also takes title of the text into account. Where the title file is not empty, the relation score of any words that occur in the chains as well as the title of the source text will be doubled. This is based on the assumption that a good title should have positive contribution to the "aboutness" of the text while at the same time, title that is too general or unrelated to the text should not affect the result of strong chains extraction.

Table 1: Score Scaling System.

	same sentence	3 sentences	same paragraph	default
synonyms	1	1	1	1
hypo/hypernym	1	0.8	0.8	0.8
antonyms	1	0.7	0.7	0.7
mero/holonym	1	0.4	0.3	0.2
siblings	1	0.2	0.1	0

There are two types of thresholds: the upper and lower thresholds.
The score of the upper threshold is calculated by:

*Upper Threshold(chain list) = average(chain scores) + 2 * standard deviation(chain scores)*

```
Algorithm for function selectNouns

initialize nounList to be an empty list of nouns/noun phrases
For each segment in text Do
    initialize segNounList to be an empty list of nouns/noun phrases in the segment
    For each sentence S in segment Do
        tokenize S
        word1 := the next token
        If word1 ends with "-" Then
            concatenate word1 with the next token
        End If
        While there are more tokens in S Do
            word2 := the next token
            If word2 ends with "-" Then
                concatenate word2 with the next token
            End If
            If word1 and word2 is a noun phrase Then
                add (word1 + word2) to segNounsList if it is not already in the list
                record the sentence number it occurs in
            Else
                If word1 is a noun Then
                    add word1 to segNounsList if it is not already in the list
                    record the sentence number it occurs in
                If word2 is a noun Then
                    add word2 to segNounsList if it is not already in the list
                    record the sentence number it occurs in
            End If
        End While
    End For
    add segNounList to nounList
End For
initialize candidateList as an empty list of candidate words
For each segment noun list in nounList Do
    For each noun/noun phrase in the segment noun list Do
        W := noun phrase/noun
        sentenceList := a list of sentences the W occurs in
        senseList := a list of synonyms, antonyms, hypernyms, and meronyms of W
        relationList := initialize to be an empty list
        CW := a new candidate word created from W, sentenceList, senseList and relationList
        add CW to candidateList
    End For
End For
return candidateList
```

Figure 1: Algorithm for function *selectNouns*.

Algorithm for function *segmentChainBuilder*

```
1   initialize unitList to be an empty list of units
2   For each candidate word CW in nounList Do
3           related_units := getRelatedUnits (CW)
4           If related_units is empty Then
5                   create a new unit with only CW in it and add this new unit to unitList
6           Else
7                   remove related_units from unitList
8                   interpList := cartesian product for all interpretations in related_units by
                        calling joinUnits (related_units)
9                   initialize newInterpList to be an empty list of interpretations
10                  For each interp in interpList Do
11                          get new lists of interpretations after trying to fit CW in interp
                            by calling fitCW(CW, interp)
12                          add these new lists to newInterpList
13                  End For
14                  choose the most active interpretations in newInterpList based on the
                      interpretation score
15                  create a new unit that contains these interpretations
16                  add the new unit to unitList
17          End If
18  End For
19  initialize chainList to be an empty list of chains
20  For each unit in unitList Do
21          get the interpretation with the highest interpretation score
22          add all the chains in this interpretation to chainList
23  End For
24  return chainList
```

Figure 2: Algorithm for function *segmentChainBuilder*.

The score of the lower threshold is simply the average of all chain scores:

$$\textit{Lower Threshold(chain list) = average(chain scores)}$$

And strong chains are chains which have:

$$\textit{score(chain)} > \textit{upper threshold (chain list)}$$

If no chains have chain score above the upper threshold, we then define our strong chains as chains which have:

$$\textit{score (chain)} > \textit{lower threshold (chain list)}$$

If still no strong chains can be found, all the chains which have a score > 0 are returned.

6 The Heuristics

Once the process of lexical chain computing has completed and strong chains have been chosen, we are ready to extract sentences from the text and form a summary. One issue regarding on the length of summary is that, should it be fixed or should it let the users choose how long they want the summary to be? In this paper, three different heuristics for extracting summary (one fixed and two flexible) have been proposed and a comparison of their performance is discussed in the next section.

6.1 Heuristic 1

Length: flexible

Algorithm: In this method, each sentence is associated with a sentence score. The score is calculated as the sum of relation scores of each word in the sentence that appears in the strong chains as well. Sentences are then ranked according to these scores, with the highest score being interpreted as the most important sentence in the source text. In doing so, users are allowed to choose the length of the summary according to how many sentences they want to be in the summary. The pseudo code of this algorithm is shown in Figure 3.

```
Algorithm for Heuristic 1

1  initialize rankList1 to be an empty list of sentence ranks for heuristic 1
2  For each Candidate Word CW in Strong Chain List Do
3        relation_scores := sum of scores CW has according to the types of
                            relations it has with other words
4        For each sentence S that CW appear Do
5              S_score += relation_scores
6        End For
7  End For
```

Figure 3: Algorithm for *Heuristic 1*.

6.2 Heuristic 2

Length: fixed

Algorithm: In this method, the term "strong word" is introduced. Strong words are defined as words in the strong chains that have a relation score greater than 1.5 * the average relation scores the words in the strong chains have. Once all the strong words are identified, the sentence that each strong word first appears in the source text is extracted to be part of summary. In this case, the summary produced by this method will be of fixed length, based on the number of strong words found in the strong chains and their locations in the text (on average, the length is around 5-15% of the source text's). Recall that the position_based summarization system assumes important sentences occur at the beginning of the text. By picking the sentence that the strong words first appear, the position_based factor is integrated into this word-word relation based method, aiming to produce a better quality of summary. The pseudo code of this algorithm is shown in Figure 4.

Algorithm for Heuristic 2

```
1   initialize strongWordList as an empty list of strong words
2   For each Candidate Word CW in Strong Chain List Do
3           relation_scores := sum of scores CW has according to the types of relations it
                                has with other words
4           If relation_scores > average of all relation_scores in Strong Chain List Then
5                   add CW to strongWordList
6           End If
7   End For
8   For each strong word SW in strongWordList Do
9           select the sentence that SW first appears
10          insert SW to the summary
11  End For
```

Figure 4: Algorithm for *Heuristic 2*.

6.3 Heuristic 3

Length: flexible

Algorithm: This method is a combination of heuristic 1 and heuristic 2. Once all the sentences are ranked and a fixed length summary becomes available, a merging sentence ranking system is used. All sentences that occur in the fixed length summary will be ranked first, and after that, the remaining sentences in the source text will be ordered in the same way as those from heuristic 1. This method gives the sentences a different ranking order compared to heuristic 1, depending on the summary produced by heuristic 2. The users are then allowed to choose the length of the summary according to how many sentences they want to be in the summary. The pseudo code of this algorithm is shown in Figure 5.

```
Algorithm for Heuristic 3

1  initialize rankList2 to be an empty list of sentence ranks for heuristic 3
2  add all sentences in fixed length summary into rankList2
3  For all sentences S in the order as rankList1 Do
4          add S to rankList2 if it is not part of the fixed length summary
5  End For
```

Figure 5: Algorithm for *Heuristic 3*.

7 Experiments and Results

Once the automatic text summarizer has been implemented, an evaluation is needed to be carried out in order to have a better understanding of the utility of the system. Evaluations of summarization systems can be divided into two types in general: intrinsic and extrinsic. Intrinsic evaluations focus on the system's quality, while extrinsic evaluations focus on the system's performance in a particular task [13]. In this paper, the intrinsic evaluation approach was taken.

7.1 Description of the Experiments

Five experiments were undertaken. The materials used included 38 articles (along with their titles and abstracts) from online journals such as Economist, Information Management Journal, Geographical Review, Scientific American and Western Journal of Medicine; 5 articles collected from the newspaper, and a commercial sentence-extraction summarization product from "Zentext"[1].

Experiment 1

Aim: Comparing the summaries produced by our summarizer to Zentext's using articles' abstract as "ideal" summaries (that are found from the original articles).

Resources Used: 38 journal articles with their abstracts, Zentext

Procedures:

1. Based on the length of the articles' abstract, Summary-1, Summary-3 and Zentext-Summary of the same length were extracted for each article.
2. All the candidate words for each four summaries of each article were extracted.
3. For each article, the number of candidate words in Summary-1, Summary-3 and Zentext-Summary that were related to the candidate words in abstracts were counted, and vice versa.

[1]available at *http://www.zentext.com/z_product_summarizer.html*

Experiment 2

Aim: Determine the sentences matching rate of summaries at various length with Zentext's.

Resources Used: 38 journal articles, Zentext

Procedures:

1. For each journal article, Summary-2, and Summary-1 and Summary-3 with length varied from 10% to 70% of the original length of the article were extracted.
2. For each journal article, Zentext-Summary with length varied from 10% to 70% of the length of original article, and a length same as Summary-2's were extracted.
3. The sentences matching rates at varied length between Zentext-Summary and Summary-1, Summary-2, and Summary-3 were calculated.

Experiment 3

Aim: Examine the effect of including the article title on summaries.

Resources Used: 38 journal articles along with their titles and abstracts, Zentext

Procedures:

1. Same as Experiment 1, but this time, the title of each article was considered.

Experiment 4

Aim: Direct human judgement of informativeness and coverage of the summaries.

Resources Used: 5 newspaper articles, Zentext, 8 human judges

Procedures:

1. For each newspaper article, Summary-2, and Summary-1, Summary-3 and Zentext-Summary with length of 15% of the original article were extracted.
2. All the summaries were grouped together with no repeated sentences.
3. For each sentence in the summary group, human judges were asked to give a rate from 0 - 10 (totally disagreed to totally agreed) based on their agreement with its inclusion in the summary of the article, if there was one.
4. The agreement rates for each sentence were averaged and used to calculate an average rate for each summary produced in step 1.

Experiment 5

Aim: Examine the effect of article length on system process time.

Resources used: 38 journal articles

Procedures:

1. Each article was run with our summarizer *(on Linux platform)*, and the time taken for the process was recorded.

8 Results and Analysis

This section shows the results obtained in all 5 experiments conducted. All the detailed results tables can be found in Appendix B.

The experimental results for *Experiment 1* is as shown in Table 2. Columns 1 and 2 show the average recall and precision rate for Summary-1. Columns 3 and 4 show the average recall and precision rate for Summary-3. Columns 5 and 6 show the average recall and precision rate for Zentext-Sumamry. From the results, it is seen that in 68.03% of the cases, the candidate words in the articles' abstracts were related to the candidate words found in the Zentext-Summary, while in 61.05% of the cases, the candidate words in the Zentext-Summary were related to the candidate words found in the articles' abstracts. This shows that, in this particular experiment, Zentext, a commercial summarization product outperformed the summarization system constructed in this paper, with an average of 4% higher in precision. Since the percentage difference was not huge, the results for Summary-1 and Summary-3 are extremely encouraging. Both had a recall and precision rate over 50%, with Summary-3 generally producing better summary results compared to Summary-1.

Table 2: Experimental results for *Experiment 1*.

Summary-1		**Summary-3**		**Zentext-Summary**	
Recall (%)	Precision(%)	Recall (%)	Precision(%)	Recall (%)	Precision(%)
62.51	57.18	66.47	57.23	68.03	61.05

Table 3 shows the experimental results for *Experiment 2*. It is shown from the table that the probability of the summarization system built in this paper chooses the same sentences as Zentext at a particular summary length is not high at all. The sentence matched rate was below 50% if the length of summary was less than 40% of the original text, with Summary-1 contained more sentences included in ZenText-Summary than Summary-3. Both Summary-1 and Summary-3 gave nearly identical results as the length of summary increases. This is due to the fact that Summary-3 was produced by combining Summary-2 and Summary-1. As mentioned before, the length of Summary-2 was generally about 5-15% of the original text. Hence for the first 20%, the sentences selected as summary for Summary-1 and Summary-3 could be very different, and therefore contributed to this gap of sentences matching rate. However, as the length of summary increases, the difference between their performances decreases.

One interesting observation can be seen from the results obtained in *Experiment 1* and *Experiment 2* is that, while the sentences selected as summary for both system were quite different, the context in each summary did not differ by much. This was proved by human inspections and by the fact that the difference between the recall and precision rate for both systems was not relatively large (as shown in *Experiment 1*, which is an experiment testing on the similarity of context represented by

Table 3: Experimental results for *Experiment 2*.

	Sentences matched rate (%)		
Length	**Summary-1**	**Summary-3**	**Summary-2**
10%	26.88	18.75	-
20%	38.71	35.79	-
30%	46.84	45.30	-
40%	54.89	53.77	-
50%	62.01	61.80	-
60%	66.69	66.85	-
70%	70.70	70.67	-
fixed length	-	-	15.84

articles' abstracts, summaries from this summarization system and summaries from Zentext). One possible explanation for this is that different summarization algorithms will most likely select different sentences to be included in a summary. As there is no single "right" summary and as in many cases, there are alternative sentences that can be used in the summary that will still give the same context. Summaries produced by different systems might be different in terms of the sentences used but very similar in terms of the context they represent.

Experiment 3 investigates the effectiveness of including an article title in the summarization system on the summaries produced. As seen in Table 4 and Table 5, both the recall and precision rate for Summary-1 and Summary-3 increased slightly after considering the inclusion of the article title. Although the increase rate was less than 1%, it confirmed with the assumption that: "a good title should be able to contribute to the 'aboutness' of the text while at the same time, title that is too general or unrelated to the text should not affect the final result."

Table 4: Experimental results for *Experiment 3* (1).

Summary-1			
No Title		**With Title**	
Recall (%)	Precision(%)	Recall (%)	Precision(%)
62.51	57.18	63.44	57.39

Table 5: Experimental results for *Experiment 3* (2).

Summary-3			
No Title		**With Title**	
Recall (%)	Precision(%)	Recall (%)	Precision(%)
66.47	57.23	66.56	57.53

In the analysis of results obtained in *Experiment 1* and *Experiment 3*, the issue of there is no single "right" summary is mentioned. Due to this fact, another experiment judging on the quality of the summarization system in this paper was conducted in *Experiment 4*. 8 independent human judges were asked to read 5 articles. As a result of human resources limitations, these 5 articles contain 22 sentences on average and were obtained from newspaper. Sentences selected by Summary-1, Summary-3 and Zentext-Summary were mixed together and human judges were then required to give a mark for each sentence (from 0 - 10), based purely on their agreement on its inclusion as a summary. The results are shown in Table 6. It is clear that the summaries produced by this summarization system favor human judgement as they performed significantly better compared to its performance with articles' abstracts, especially Summary-2, which achieved an agreement as high as 80.59%. Again, Summary-3 performed better than Summary-1, with an average of 74.17% agreement within the human judges. Zentext performed worse in this experiment, with an average of only 65.98% agreement within human judges. However, as only 5 articles were used, more testing has to be done to confirm the findings from this experiment, though the results obtained so far have been very encouraging. Two of these five articles are shown in Appendix A, together with all its sentences choices and summaries produced by our summarizer and Zentext.

Table 6: Experimental results for *Experiment 4*.

	Summ-1 (%)	**Summ-2 (%)**	**Summ-3 (%)**	**Zentext (%)**
Article 1	74.58	95.00	74.58	57.08
Article 2	69.38	83.75	75.94	62.50
Article 3	56.25	85.63	85.63	78.13
Article 4	69.06	76.88	67.19	58.75
Article 5	67.50	61.67	67.50	73.44
Mean	**67.35**	**80.59**	**74.17**	**65.98**

A working algorithm is not sufficient, the efficiency of the system needs to be considered as well. As a result, *Experiment 5* was conducted to determine the process time for articles of different length and investigate the efficiency and scalability of the summarization system built in this paper. The same 38 articles were used, with the shortest article containing 22 sentences and the longest article containing 389 sentences. Process time is determined by 3 factors: number of sentences, number of words and number of candidate words in each article. Figure 6, 7 and 8 show the relationship between process time and these 3 factors respectively (Time on X-axis, No. of words/sentences on Y-axis). All 3 graphs suggest a linear relationships, and hence it also suggests that the algorithm used in this paper has a linear runtime. One big factor that contributes to this is the method used for getting related senses for each word and the method used for checking the hypernym/hyponym relationship between two words in this paper. Instead of walking up and down the hypernym/hyponym graph in WordNet to collect information as Barzilay and Elhadad did [2] (which resulted in exponential efficiency as suggested by Silber and McCoy [40]), in this paper, for checking hypernym/hyponym relationship between two words, no hyponyms of words are needed. Two words are hypernym/hyponym if one is the hypernym of another one. As a result, only the hypernyms information for each word is required and stored. This has largely reduced the time for going down all possible routes in the hypernym/hyponym graph collecting all the children information of a word, as there is only one route up to get a word's parents information.

Again, the results obtained in this experiment are extremely promising. It not only suggests that the algorithm used is efficient itself, but also suggests that this system has a very high potential in its feasibility in summarizing documents of longer length.

References

[1] Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. In *Proceedings of the ACL'97/EACL'07 Workshop on Intelligent Scalable Text Summarization*, Madrid,

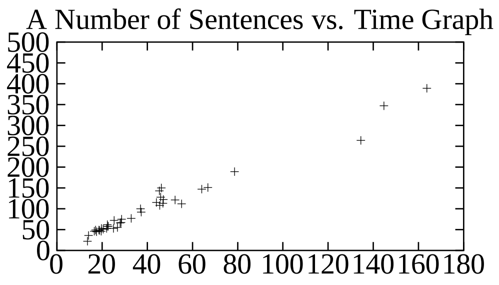

Figure 6: Graphical Representation for results in *Experiment 5* (a).

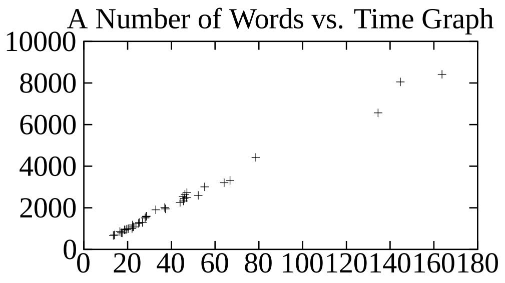

Figure 7: Graphical Representation for results in *Experiment 5* (b).

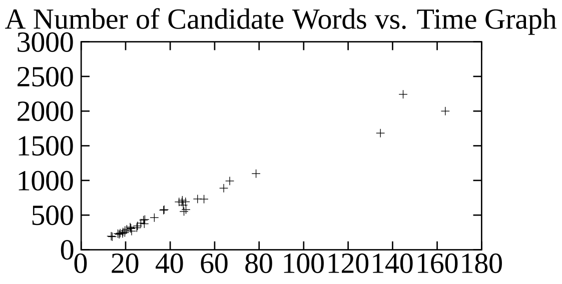

Figure 8: Graphical Representation for results in *Experiment 5* (c).

Spain, July 11 1997.

[2] Regina Barzilay and Michael Elhadad. Lexical chains for summarization. Master Thesis, Department of Mathematics and Computer Science, Ben-Gurion University of the Negev, 1997.

[3] M. Brunn, Y. Chali and C. J. Pinchak. *Text summarization using lexical chains*. Department of Mathematics and Computer Science, University of Lethbridge, 2000.

[4] Freddy Y. Y. Choi. Advances in domain independent linear text segmentation. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics*, pages 26-33, Seattle, Washington, 2000.

[5] H. P. Edmundson. New methods in automatic abstracting. *Journal of the ACM*, 16(2):264-285, 1969.

[6] Steohen J. Green. Building hypertext links in newspaper articles using semantic similarity. In *Proceedings of the Third Workshop on Applications of Natural Language to Information Systems (NLDB'97)*, pages 178-190, Vancouver, British Columbia, June 1997.

[7] Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175-204, July-September 1986.

[8] Michael Hasan and Ruqaiya Halliday. *Cohesion in English*. Longman, London, 1976.

[9] M. Hearst. Multi-paragraph segmentation of expository text. In *Proceedings of the 32th Annual Meeting of the Association for Computational Linguistics (ACL-94)*, pages 9-16, Las Cruces, New Mexico, 1994.

[10] Gramme Hirst and David St-Onge. Lexical chains as representation of context for the detection and correction of malapropisms. In Christiane Fellbaum, editor, *WordNet: An electronic lexical database and some of its applications*. The MIT Press, Cambridge, 1997.

[11] Eduard H.Hovy and Chin Yew Lin. Automated text summarization in SUMMARIST. In *Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*, pages 18-24, Madrid, Spain, July 11 1997.

[12] H. Jing. Sentence reduction for automatic text summarization. In *Proceedings of ANLP*, 2000.

[13] H. R. Jing, R. Barzilay, K. McKeown and M. Elhadad. Summarization evaluation methods: Experiments and analysis. In *Working Notes of the AAAI Spring Symposium on Intelligent Text Summarization*, pages 60-68. Menlo Park, Calif., AAAI Press, Spring 1998.

[14] H. Jing and Kahleen R. McKeown. The decomposition of human-written summary sentences. In *Proceedings of the 22nd International ACM SIGIR Congerence on Research and Development in Information Retrieval (SIGIR'99)*, pages 129-136, University of Berkeley, CA, August, 1999.

[15] H. Jing and K. McKeown. Cut and paste based text summarization. In *Proceedings of NAACL'00*, Seattle, WA, 2000.

[16] Rick Kazman, Reem Al-Halimi, William Hunt, and Marilyn Mantei. Four paradigms for indexing video conferences. *IEEE MultiMedia*, 1996.

[17] C. Kennedy and B. Boguraev. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *Proceedings of the 16th International Conference on Computational Linguistics*, 1996.

[18] Hideki Kozima and Akira Ito. Context-sensitive [measurement of] word distance by adaptive scaling of a semantic space. In Ruslan Mitkov and Nicolas Nicolov, editors, *Recent Advances in Natural Language Processing: Selected Papers from RANLP'95*, volume 136 of *Amsterdam Studies in the Theory and History of Linguistic Science: Current Issues in Linguistic Theory*, chapter 2, pages 111-124, John Benjamins Publishing Company, Amsterdam/Philadelphia, 1997.

[19] Julian Kupiec, Jan Pederson, and Francine Chen. A trainable document summarizer. In *SIGIR '95*, pages 68-73, Seattle, Washington, 1995.

[20] S. Lappin and H. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4), 1994.

[21] Claudia Leacock and Martin Chodorow. Combining local context and WordNet similarity for word sense identification. In Christiane Fellbaum, editor, *WordNet: An Electronic Lexical Database*, chapter 11, pages 265-283. The MIT Press, Cambridge, MA, 1998.

[22] Dekang Lin. Using syntactic dependency as local context to resolve word sense ambiguity. In *Proceedings of ACL/EACL-97*, pages 64-71, Madrid, Spain, July 1997.

[23] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159-165, April 1958.

[24] Inderjeet Mani, Barbara Gates, and Eric Bloedorn. Improving summaries by revising them. In *Proceedings of the 37th Annual Conference of the Association for Computational Linguistics*, College Park, MD. ACL, 1999.

[25] Inderjeet Mani and Mark T. Maybury, editors. A*dvances in automatic text summarization*. Cambridge, Mass. : MIT Press, 1999.

[26] Willian C. Mann and Sandra A. Thompson. Rhetorical structure theory: toward a functional theory of text organization. *Text*, 8(3):243-281, 1988.

[27] Daniel Marcu. The Rhetorical Parsing, Summarization, and Generation of Natural Language Text. Ph.D thesis, Department of Computer Science, University of Toronto, December, 1997.

[28] M. T. Maybury. Automated event summarization techniques. In *Summarizing Text for Intelligent Communication*, 101-149, 1995.

[29] Kathleen R. McKeown and Dragomir R. Radev. Generating summaries of multiple news articles. In *Proceedings of the Seventeenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 74-82, Seattle, Washington, 1995.

[30] George A. Miller, Richard Beckwith, Christiane Fellaum, Derek Gross, and Katherine J. Miller. Introduction to WordNet. An online lexical database. *International Journal of Lexicography (special issue)*, 3(4):235-312, 1990.

[31] J. Morris and G. Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of the text. *Computational Linguistics*, 17(1):21-45, 1001.

[32] Manabu Okumura and Takeo Honda. Word sense disambiguation and text segmentation based on lexical cohesion. In *Proceedings of the Fifteenth International Conference on Computational Linguistics (COLING-94)*, volume 2, pages 755-761, Kyoto, Japan, August 1994.

[33] Kenji Ono, Kazuo Sumita, and Seiji Miike. Abstract generation base on rhetorical structure extraction. In *Proceedings of the Internaltional Conference on Computational Linguistics (Coling-94)*, pages 344-348, Japan, 1994.

[34] C.D. Paice. The automatic generation of literature abstracts: an approach based on the identification of self-indicating phrases. In R. N. Oddy, S. E. Robertson, C. J. van Rijsbergen, and P.W. Williams, editors, *Information Retrieval Research*, pages 172-191, Butterworths, 1981.

[35] C.D Paice. Constructing literature abstracts by computer: techniques and prospects. *Information Processing and Management*, 26(1):171-186, 1990.

[36] Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics,* 19(1):17-30, February 1989.

[37] Jeffrey C. Reynar and Adwait Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, D.C., 1997.

[38] Ray Richardson and Allan F. Smeaton. Using WordNet in a knowledge-based approach to information retrieval Working paper CA-0395, School of Computer Applications, Dublin City University, 1995.

[39] Gerard Salton and Amit Singhal. Automatic text decomposition and structuring. *Information Processing and Management*, 32(2):127-138, 1996.

[40] H. Gregory Silber and Kathleen F. McCoy. Efficient text summarization using lexical chains. In *2000 International Conference on Intelligent User Interfaces*, New Orleans, LA, January, 2000.

[41] Michael Sussna. Word sense disambiguation for free-text indexing using a massive semantic network. In *Proceedings of the Second International Conference on Information and Knowledge Management (CIKM-93)*, pages 67-74, Arlington, Virginia, 1993.

[42] Simone Teufel and Marc Moens. Sentence extraction as a classification task. In *Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization*, pages 58-65, Madrid, Spain, July 11 1997.

[43] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 133-138, Las Cruces, New Mexico, June 1994.