An Approach to Text Summarization

Sankar K
AU-KBC Research Centre
MIT Campus, Anna University
Chennai- 44.
sankar@au-kbc.org

Sobha L
AU-KBC Research Centre
MIT Campus, Anna University
Chennai- 44.
sobha@au-kbc.org

Abstract

We propose an efficient text summarization technique that involves two basic operations. The first operation involves finding coherent chunks in the document and the second operation involves ranking the text in the individual coherent chunks and picking the sentences that rank above a given threshold. The coherent chunks are formed by exploiting the lexical relationship between adjacent sentences in the document. Occurrence of words through repetition or relatedness by sense relation plays a major role in forming a cohesive tie. The proposed text ranking approach is based on a graph theoretic ranking model applied to text summarization task.

1 Introduction

Automated summarization is an important area in NLP research. A variety of automated summarization schemes have been proposed recently. NeATS (Lin and Hovy, 2002) is a sentence position, term frequency, topic signature and term clustering based approach and MEAD (Radev et al., 2004) is a centroid based approach. Iterative graph based Ranking algorithms, such as Kleinberg's HITS algorithm (Kleinberg, 1999) and Google's PageRank (Brin and Page, 1998) have been traditionally and successfully used in web-link analysis, social networks and more recently in text processing applications (Mihalcea and Tarau, 2004), (Mihalcea et al., 2004), (Erkan and Radev, 2004) and (Mihalcea, 2004). These iterative approaches have a high time complexity and are practically slow in dynamic summarization. Proposals are also made for coherence based automated summarization system (Silber and McCoy, 2000).

We propose a novel text summarization technique that involves two basic operations, namely finding coherent chunks in the document and ranking the text in the individual coherent chunks formed.

For finding coherent chunks in the document, we propose a set of rules that identifies the connection between adjacent sentences in the document. The connected sentences that are picked based on the rules form coherent chunks in the document. For text ranking, we propose an automatic and unsupervised graph based ranking algorithm that gives improved results when compared to other ranking algorithms. The formation of coherent chunks greatly improves the amount of information of the text picked for subsequent ranking and hence the quality of text summarization.

The proposed text ranking technique employs a hybrid approach involving two phases; the first phase employs word frequency statistics and the second phase involves a word position and string pattern based weighing algorithm to find the weight of the sentence. A fast running time is achieved by using a compression hash on each sentence.

Proceedings of CLIAWS3, Third International Cross Lingual Information Access Workshop, pages 53–60,
Boulder, Colorado, June 2009.

This paper is organized as follows: section 2 discusses lexical cohesion, section 3 discusses the text ranking algorithm and section 4 describes the summarization by combining lexical cohesion and summarization.

2 Lexical Cohesion

Coherence in linguistics makes the text semantically meaningful. It is achieved through semantic features such as the use of deictic (a deictic is an expression which shows the direction. ex: that, this.), anaphoric (a referent which requires an antecedent in front. ex: he, she, it.), cataphoric (a referent which requires an antecedent at the back.), lexical relation and proper noun repeating elements (Morris and Hirst, 1991). Robert De Beaugrande and Wolfgang U. Dressler define coherence as a "continuity of senses" and "the mutual access and relevance within a configuration of concepts and relations" (Beaugrande and Dressler, 1981). Thus a text gives meaning as a result of union of meaning or senses in the text.

The coherence cues present in a sentence are directly visible when we go through the flow of the document. Our approach aims to achieve this objective with linguistic and heuristic information.
The identification of semantic neighborhood, occurrence of words through repetition or relatedness by sense relation namely *synonyms*, *hyponyms* and *hypernym*, plays a major role in forming a cohesive tie (Miller et al., 1990).

2.1 Rules for finding Coherent chunks

When parsing through a document, the relationship among adjacent sentences is determined by the *continuity* that exists between them.

We define the following set of rules to find coherent chunks in the document.

Rule 1

The presence of *connectives* (such as *accordingly, again, also, besides)* in present sentence indicates the connectedness of the present sentence with the previous sentence. When such *connectives* are found, the adjacent sentences form coherent chunks.

Rule 2

A 3rd person pronominal in a given sentence refers to the antecedent in the previous sentence, in such a way that the given sentence gives the complete meaning with respect to the previous sentence. When such adjacent sentences are found, they form coherent chunks.

Rule 3

The reappearance of NERs in adjacent sentences is an indication of connectedness. When such adjacent sentences are found, they form coherent chunks.

Rule 4

An ontology relationship between words across sentences can be used to find semantically related words across adjacent sentences that appear in the document. The appearance of related words is an indication of its coherence and hence forms coherent chunks.
All the above rules are applied incrementally to achieve the complete set of coherent chunks.

2.1.1 Connecting Word

The ACE Corpus was used for studying the coherence patterns between adjacent sentences of the document. From our analysis, we picked out a set of keywords such that, the appearance of these keywords at the beginning of the sentence provide a strong lexical tie with the previous sentence.
The appearance of the keywords *"accordingly, again, also, besides, hence, henceforth, however, incidentally, meanwhile, moreover, namely, nevertheless, otherwise, that is, then, therefore, thus, and, but, or, yet, so, once, so that, than, that, till, whenever, whereas and wherever"*, at the beginning of the present sentence was found to be highly coherent with the previous sentence.

Linguistically a sentence cannot start with the above words without any related introduction in the previous sentence.

Furthermore, the appearance of the keywords "*consequently*, *finally*, *furthermore*", at the beginning or middle of the present sentence was found to be highly cohesive with the previous sentence.
Example 1

1. a The train was late.
1. b *However* I managed to reach the wedding on time.

In Example 1, the connecting word *however* binds with the situation of the train being late.

Example 2

1. a The cab driver was late.
1. b The bike tyre was punctured.
1. c The train was late.
1 .d *Finally*, I managed to arrive at the wedding on time by calling a cab.

Example 3

1. a The cab driver was late.
1. b The bike tyre was punctured.
1. c The train was late.
1. d I could not wait any more; I *finally* managed to reach the wedding on time by calling a cab.

In Example 2, the connecting word *finally* binds with the situation of him being delayed. Similarly, in Example 3, the connecting word *finally*, though it comes in the middle of the sentence, it still binds with the situation of him being delayed.

2.1.2 Pronominals

In this approach we have a set of pronominals which establishes coherence in the text. From our analysis, it was observed that if the pronominals *"he, she, it, they, her, his, hers, its, their, theirs"*, appear in the present sentence; its antecedent may be in the same or previous sentence.

It is also found that if the pronominal is not possessive (i.e. the antecedent appears in the previous sentence or previous clause), then the present sentence and the previous sentences are connected. However, if the pronominal is possessive then it behaves like reflexives such as "himself", "herself" which has subject as its antecedent. Hence the possibility of connecting it with the previous sentence is very unlikely. Though pronominal resolution cannot be done at a window size of 2 alone, still we are looking at window size 2 alone to pick guaranteed connected sentences.

Example 4

1. a *Ravi* is a good boy.
1. b *He* always speaks the truth.

In Example 4, the pronominal he in the second sentence refers to the antecedent *Ravi* in the first sentence.

Example 5

1. a *He* is the one who got the first prize.

In example 5 the pronominal he is possessive and it doesn't need an antecedent to convey the meaning.

2.1.3 NERs Reappearance

Two adjacent sentences are said to be coherent when both the sentences contain one or more reappearing nouns.

Example 6

1. a *Ravi* is a good boy.
1. b *Ravi* scored good marks in exams.

Example 7

1. a The *car* race starts at noon.
1. b Any *car* is allowed to participate.

Example 6 and Example 7 demonstrates the coherence between the two sentences through reappearing nouns.

2.1.4 Thesaurus Relationships

WordNet covers most of the sense relationships. To find the semantic neighborhood between adjacent sentences, most of the lexical relationships such as synonyms, hyponyms, hypernyms, meronyms, holonyms and gradation can be used (Fellbaum 1998). Hence, semantically related terms are captured through this process.

Example 8

1. a The *bicycle* has two wheels.
1. b The *wheels* provide speed and stability.

In Example 8, *bicycle* and wheels are related through *bicycle* is the *holonym* of *wheels*.

2.2 Coherence Finding Algorithm

The algorithm is carried out in four phases. Initially, each of the 4 cohesion rules is individually applied over the given document to give coherent chunks. Next, the coherent chunks obtained in each

phases are merged together to give the global coherent chunks in the document.

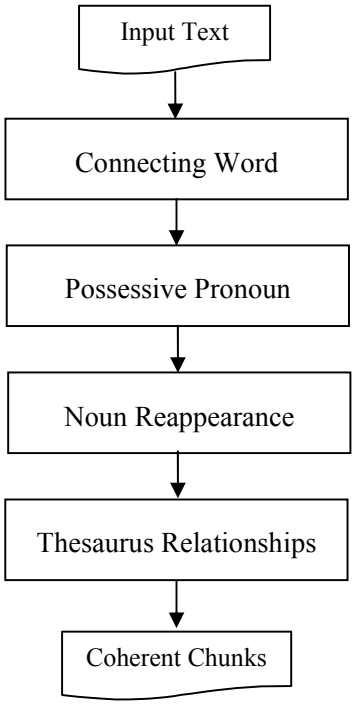

Figure 1: Flow of Coherence chunker

Figure 1, shows the flow and rule positions in the coherence chunk identification module.

2.3 Evaluation

One way to evaluate the coherence finding algorithm is to compare against human judgments made by readers, evaluating against text pre marked by authors and to see the improved result in the computational task. In this paper we will see the computational method to see the improved result.

3 Text Ranking

The proposed graph based text ranking algorithm consists of three steps: (1) Word Frequency Analysis; (2) A word positional and string pattern based weight calculation algorithm; (3) Ranking the sentences by normalizing the results of step (1) and (2).

The algorithm is carried out in two phases. The weight metric obtained at the end of each phase is averaged to obtain the final weight metric. Sentences are sorted in non ascending order of weight.

3.1 Graph

Let G (V, E) be a weighted undirected complete graph, where V is set of vertices and E is set of weighted edges.

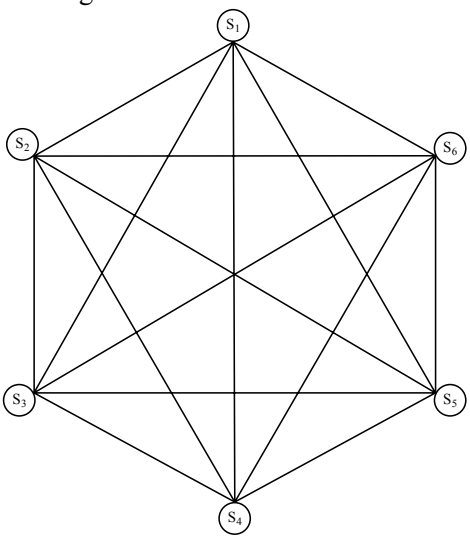

Figure 2: A complete undirected graph

In figure 2, the vertices in graph G represent the set of all sentences in the given document. Each sentence in G is related to every other sentence through the set of weighted edges in the complete graph.

3.2 Phase 1

Let the set of all sentences in document $S = \{s_i \mid 1 \le i \le n\}$, where n is the number of sentences in S. The sentence weight (SW) for each sentence is calculated by average affinity weight of words in it. For a sentence $s_i = \{w_j \mid 1 \le j \le m_i\}$ where m_i is the number of words in sentence s_i, $(1 \le i \le n)$ the affinity weight AW of a word w_j is calculated as follows:

$$AW(w_j) = \frac{\sum_{\forall w_k \in S} IsEqual(w_j, w_k)}{WC(S)} \qquad (1)$$

where S is the set of all sentences in the given document, w_k is a word in S, WC (S) is the total number of words in S and function $IsEqual(x, y)$ returns an integer count 1 if x and y are equal else integer count 0 is returned by the function.

Next, we find the sentence weight $SW(s_i)$ of each sentence s_i $(1 \le i \le n)$ as follows:

$$SW(s_i) = \frac{1}{m_i} \sum_{\forall w_j \in s_i} AW(w_j) \quad (2)$$

At the end of phase 1, the graph vertices hold the sentence weight as illustrated in figure 4.

[1]"The whole show is dreadful," she cried, coming out of the menagerie of M. Martin.
[2]She had just been looking at that daring speculator "working with his hyena" to speak in the style of the program.
[3]"By what means," she continued, "can he have tamed these animals to such a point as to be certain of their affection for."
[4]"What seems to you a problem," said I, interrupting, "is really quite natural."
[5]"Oh!" she cried, letting an incredulous smile wander over her lips.
[6]"You think that beasts are wholly without passions?" Quite the reverse; we can communicate to them all the vices arising in our own state of civilization.

Figure 2: Sample text taken for the ranking process.

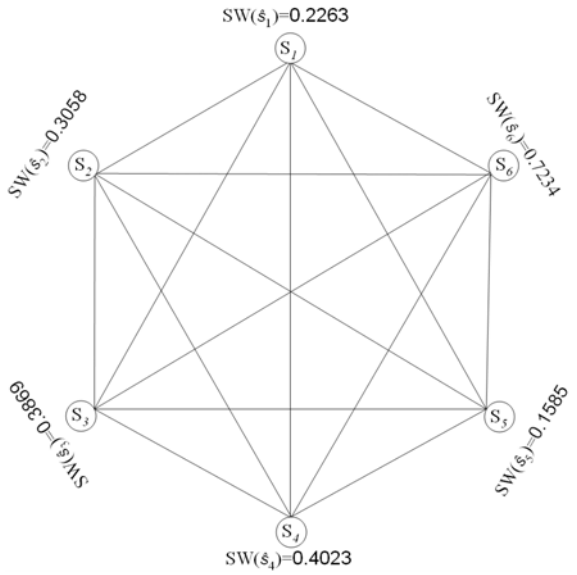

Figure 4: Sample graph of Sentence weight calculation in phase 1

3.3 Compression hash

A fast compression hash function over word w is given as follows:

$$H(w) = (c_1 a^{k-1} + c_2 a^{k-2} + c_3 a^{k-3} + \ldots + c_k a^0) \bmod p \quad (3)$$

where $w = \{c_1, c_2, c_3 \ldots c_k\}$ is the ordered set of ASCII equivalents of alphabets in w and k the total number of alphabets in w. The choice of a=2 permits the exponentiations and term wise multiplications in equation 3 to be binary shift operations on a micro processor, thereby speeding up the hash computation over the text. Any lexicographically ordered bijective map from character to integer may be used to generate set w. The recommendation to use ASCII equivalents is solely for implementation convenience. Set p = 26 (for English), to cover the sample space of the set of alphabets under consideration.

Compute $H(w)$ for each word in sentence s_i to obtain the hashed set

$$H(s_i) = \{H(w_1), H(w_2) \ldots H(w_{mi})\} \quad (4)$$

Next, invert each element in the set H (s_i) back to its ASCII equivalent to obtain the set

$$\hat{H}(\hat{s}_i) = \{H(\hat{c}_1), H(\hat{c}_2) \ldots H(\hat{c}_{mi})\} \quad (5)$$

Then, concatenate the elements in set $\hat{H}(s_i)$ to obtain the string $\hat{s}_i$; where $\hat{s}_i$ is the compressed representation of sentence s_i. The hash operations are carried out to reduce the computational complexity in phase 2, by compressing the sentences and at the same time retaining their structural properties, specifically word frequency, word position and sentence patterns.

3.4 Levenshtein Distance

Levenshtein distance (*LD*) between two strings *string*1 and *string*2 is a metric that is used to find the number of operations required to convert *string*1 to *string*2 or vice versa; where the set of possible operations on the character is insertion, deletion, or substitution.

The *LD* algorithm is illustrated by the following example

LD (ROLL, ROLE) is 1
LD (SATURDAY, SUNDAY) is 3

3.5 Levenshtein Similarity Weight

Consider two strings, *string*1 and *string*2 where ls_1 is the length of *string*1 and ls_2 be the length of *string*2. Compute *MaxLen*=maximum (ls_1, ls_2). Then *LSW* between *string*1 and *string*2 is the difference between *MaxLen* and *LD*, divided by *MaxLen*. Clearly, *LSW* lies in the interval 0 to 1. In case of a perfect match between two words, its *LSW* is 1 and in case of a total mismatch, its *LSW* is 0. In all other cases, 0 < *LSW* <1. The *LSW* metric is illustrated by the following example.

LSW (ABC, ABC) =1
LSW (ABC, XYZ) =0
LSW (ABCD, EFD) =0.25

Hence, to find the Levenshtein similarity weight, first find the Levenshtein distance *LD* using which *LSW* is calculated by the equation

$$LSW(\hat{s}_i, \hat{s}_j) = \frac{MaxLen(\hat{s}_i, \hat{s}_j) - LD(\hat{s}_i, \hat{s}_j)}{MaxLen(\hat{s}_i, \hat{s}_j)} \qquad (6)$$

where, $\hat{s}_i$ and $\hat{s}_j$ are the concatenated string outputs of equation 5.

3.6 Phase 2

Let $S = \{s_i \mid 1 \le i \le n\}$ be the set of all sentences in the given document; where n is the number of sentences in S. Further, $s_i = \{w_j \mid 1 \le j \le m\}$, where m is the number of words in sentence s_i.

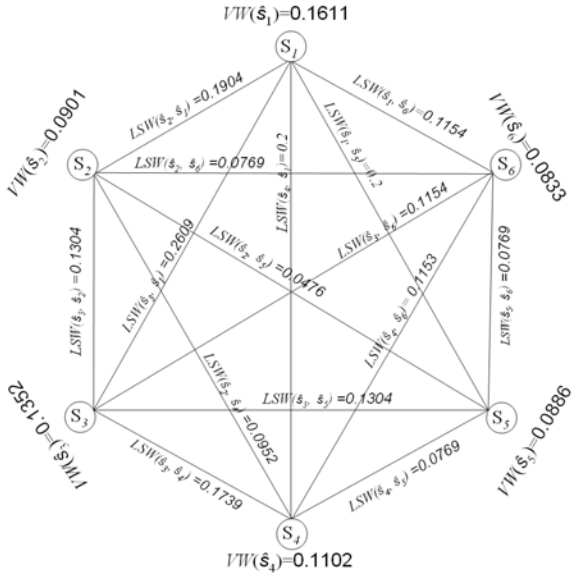

Figure 5: Sample graph for Sentence weight calculation in phase 2

$\forall s_i \in S$,find $\hat{H}(\hat{s}_i) = \{H(\hat{c}_1), H(\hat{c}_2)...H(\hat{c}_{mi})\}$ using equation 3 and 4. Then, concatenate the elements in set $\hat{H}(s_i)$ to obtain the string $\hat{s}_i$; where $\hat{s}_i$ is the compressed representation of sentence s_i.

Each string $\hat{s}_i$; $1 \le i \le n$ is represented as the vertex of the complete graph as in figure 5 and $\hat{S}=\{\hat{s}_i \mid 1 \le i \le n\}$. For the graph in figure 5, find the Levenshtein similarity weight *LSW* between every vertex using equation 6. Find vertex weight (*VW*) for each string $\hat{s}_i$; $1 \le l \le n$ by

$$VW(\hat{s}_l) = \frac{1}{n} \sum_{\forall \hat{s}_i \ne \hat{s}_l \in \hat{S}} LSW(\hat{s}_l, \hat{s}_i) \qquad (7)$$

4 Text Ranking

The rank of sentence s_i; $1 \le i \le n$ is computed as

$$Rank(s_i) = \frac{SW(s_i) + VW(\hat{s}_i)}{2}; 1 \le i \le n \qquad (8)$$

where, $SW(s_i)$ is calculated by equation 2 of phase 1 and $VW(\hat{s}_i)$ is found using equation 7 of phase 2. Arrange the sentences s_i; $1 \le i \le n$, in non increasing order of their ranks.

$SW(s_i)$ in phase 1 holds the sentence affinity in terms of word frequency and is used to determine the significance of the sentence in the overall raking scheme. $VW(\hat{s}_i)$ in phase 2 helps in the overall ranking by determining largest common subsequences and other smaller subsequences then assigning weights to it using *LSW*. Further, since named entities are represented as strings, repeated occurrences are weighed efficiently by *LSW*, thereby giving it a relevant ranking position.

5 Summarization

Summarization is done by applying text ranking over the global coherent chunks in the document. The sentences whose weight is above the threshold is picked and rearranged in the order in which the sentences appeared in the original document.

6 Evaluation

The ROUGE evaluation toolkit is employed to evaluate the proposed algorithm. ROUGE, an automated summarization evaluation package based on Ngram statistics, is found to be highly correlated with human evaluations (Lin and Hovy, 2003a).

The evaluations are reported in ROUGE-1 metrics, which seeks unigram matches between the generated and the reference summaries. The ROUGE-1 metric is found to have high correlation with human judgments at a 95% confidence level and hence used for evaluation. (Mihalcea and Tarau, 2004) a graph based ranking model with Rouge score 0.4904, (Mihalcea, 2004) Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization with Rouge score 0.5023.

Table 1 shows the ROUGE Score of 567 news articles provided during the Document Understanding Evaluations 2002(DUC, 2002) using the proposed algorithm without the inclusion of coherence chunker module.

	Score
ROUGE-1	0.5103
ROUGE-L	0.4863

Table 1: ROUGE Score for the news article summarization task without coherence chunker, calculated across 567 articles.

Table 2 shows the ROUGE Score of 567 news articles provided during the Document Understanding Evaluations 2002(DUC, 2002) using the proposed algorithm after the inclusion of coherence chunker module.

	Score
ROUGE-1	0.5312
ROUGE-L	0.4978

Table 2: ROUGE Score for the news article summarization task with coherence chunker, calculated across 567 articles.

Comparatively Table 2, which is the the ROUGE score for summary including the coherence chunker module gives better result.

7 Related Work

Text extraction is considered to be the important and foremost process in summarization. Intuitively, a hash based approach to graph based ranking algorithm for text ranking works well on the task of extractive summarization. A notable study report on usefulness and limitations of automatic sentence extraction is reported in (Lin and Hovy, 2003b), which emphasizes the need for efficient algorithms for sentence ranking and summarization.

8 Conclusions

In this paper, we propose a coherence chunker module and a hash based approach to graph based ranking algorithm for text ranking. In specific, we propose a novel approach for graph based text ranking, with improved results comparative to existing ranking algorithms. The architecture of the algorithm helps the ranking process to be done in a time efficient way. This approach succeeds in grabbing the coherent sentences based on the linguistic and heuristic rules; whereas other supervised ranking systems do this process by training the summary collection. This makes the proposed algorithm highly portable to other domains and languages.

References

ACE Corpus. NIST 2008 Automatic Content Extraction Evaluation(ACE08). *http://www.itl.nist.gov/iad/mig/tests/ace/2008/*

Brin and L. Page. 1998. The anatomy of a large scale hypertextualWeb search engine. *Computer Networks and ISDN Systems,* 30 (1 – 7).

Erkan and D. Radev. 2004. Lexpagerank: Prestige in multi document text summarization. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing,* Barcelona, Spain, July.

Fellbaum, C., ed. WordNet: An electronic lexical database. *MIT Press, Cambridge* (1998).

Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604-632.

Lin and E.H. Hovy. From Single to Multi document Summarization: A Prototype System and its Evaluation. *In Proceedings of ACL-2002*.

Lin and E.H. Hovy. 2003a. Automatic evaluation of summaries using n-gram co-occurrence statistics. *In Proceedings of Human Language Technology Conference (HLT-NAACL 2003),* Edmonton, Canada, May.

Lin and E.H. Hovy. 2003b. The potential and limitations of sentence extraction for summarization. *In Proceedings of the HLT/NAACL Workshop on Automatic Summarization,* Edmonton, Canada, May.

Mihalcea. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. *In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004) (companion volume)*, Barcelona, Spain.

Mihalcea and P. Tarau. 2004. TextRank - bringing order into texts. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, Barcelona, Spain.

Mihalcea, P. Tarau, and E. Figa. 2004. PageRank on semantic networks, with application to word sense disambiguation. *In Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004),* Geneva, Switzerland.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K. J. Introduction to WordNet: An on-line lexical database. *Journal of Lexicography* (1990).

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

Radev, H. Y. Jing, M. Stys and D. Tam. Centroid-based summarization of multiple documents. *Information Processing and Management,* 40: 919-938, 2004.

Robert de Beaugrande and Wolfgang Dressler. Introduction to Text Linguistics. *Longman*, 1981.

Silber, H. G., McCoy, K. F. Efficient text summarization using lexical chains. *In Proceedings of Intelligent User Interfaces*. (2000).