Journal of Computer Science & Its Applications, June 2007, Vol. 14 No. 1

Tool Support for Content Summarization Based on Lexical Chain Approach Department of Computer and Information Sciences

Olugbara, O.O., Daramola, J.O., Apata, O.O.
Covenant University, Ota, Ogun State, Nigeria.
{olu_olugbara, wandi_ex}@yahoo.com, omomide@yahoo.co.uk

ABSTRACT

The subject of automatic content summarization has attracted a lot of keen interest for a long time. This is because of the necessity to quickly mine useful summaries from the vast and ever-increasing volume of information being made available from disparate electronic sources including the World Wide Web. This situation accentuates the need for adequate software tool support for content summarization in order to widen the distribution and accessibility to this important functionality. In this paper, we report the implementation of an ActiveX server component for content summarization based on improved lexical chain approach. We have included in our implementation a local dictionary to complement WordNet and a simple rule-based part-of-speech tagger in order to improve the selection of candidate words for summary extraction. This tool has a GUI interface, for learning new words and usage senses and so naturally possesses the attribute to improve with use and is able to handle language domain peculiarities.

Keywords: Content summarization, Lexical chain, ActiveX Component

1. Introduction

The advent of the information technology age has made electronic documentation to become the de-facto standard media for business, social and academic information. Today, there exist enormous and ever-increasing volumes of information in various electronic document media, which makes automatic content summarization highly desirable.

Automatic Content Summarization (ACS) entails a procedure for automatically extracting or generating content summary from a document taking into cognizance the topical domain of discussion in a document. There exists a wide field of application areas for content summarization that makes adequate software tool support very essential. For example with the amount of information available on the Internet, an automatic summarizer in combination with a search agent will reduce the time to scout for useful information. This also includes content reduction of website information when downloaded to mobile devices with limited memories, automatic document abstracting, generation of newspaper articles summary, and summarization of multi-documents. Content summarization tools will tremendously aid qualitative data analysis. Qualitative Data Analysis (QDA) is a challenging area whose goal is to summarise message contents which are common words, phrases, themes or patterns that can be seen, heard, or read in books, written documents, brochures, speeches, transcripts, new articles and visual media. QDA has useful applications in

journalism, legal practice, forensic, health care delivery, library services and education.

The objective of this work is to give the description of the underlining algorithms and concepts employed in the implementation of an ActiveX component for text summarization. The algorithm derives considerably from the models presented in [1],[10], but with some improvements so as to accommodate domain language peculiarities

Firstly, we have included a local customizable word base to complement the WordNet lexical database used in [2] for the identification of proper nouns such as localized names of things and places that are not part of WordNet database.

Secondly, we introduce a learning interface for learning new words, usage context and senses for proper adaptation to its usage environment. These enhancements has brought improvements to the process of selecting candidate words that can affect the aboutness of a document

Thirdly, our algorithm for computing lexical chains is based on strong and direct relationships among words unlike the extra-strong, strong and medium strong relationships used in [10]. This comes with inherent benefit of reducing the ambiguity to the minimum.

The overall layout of the rest part of the paper is as follows. In section 2, we provide background information on previous related works. In section 3, we present the methodology adopted in implementing the described algorithm and the process of packaging it as a tool. Section 4 is a discussion based on the evaluation of the performance of the tool using empirical tests and the paper closes in section 5 with the highlight of future work direction.

2. Background Information

Automatic summarization was first attempted in the 1950s, as auto-extracts [13]. However, the increasing volume of machine-readable text, and advances in natural language processing, have stimulated a new interest in automatic summarizing.

In automatic text summarization, the two distinct techniques that are available are text extraction and text abstraction. Text extraction entails mining of pieces of an original text on a statistical basis or with heuristics into a shorter text with the original information content preserved. Text abstraction is the process of parsing the original text in a linguistic way, interpreting it and finding new concepts to describe the text and generating a new shorter text with the same information content. The latter is very similar to text generation. However, majority of research work carried out in text summarization makes use of text extraction for summary generation from source representation as a result of the interpretation of the text.

Research efforts in text summarization can also be classified into single document summarization and multiple document summarization based on the type of input. Single document summarization involves the summarization of a single document while multiple documents summarization involves the summarization from multiple independent documents discussing the same subject. However, most research work in automatic text summarization has been towards single document summarization, since multiple documents can be combined into single documents.

Early works in this direction include [13], Position based method [12], Rhetorical parsing [15].

Research using lexical chains and related techniques has received much attention over the years. This was first introduced in [17], where they represented cohesion among related terms within a text. The authors suggested that the discourse structure of a text may be determined by finding lexical chains in the text, where a lexical chain is a cohesive chain in which the criterion for inclusion of a word is that it bears some kind of cohesive relationship to a word that is already in the chain. They based the building of the lexical chain on an abridged version of the Roget's Thesaurus, identifying five basic classes of dependency relationships. However, they could not implement their algorithm for lack of a computer readable form of the thesaurus. Later work using lexical chains was conducted in [9] using lexical chains to correct malapropism and WordNet, a lexical database for some semantic information as the knowledge source. They used three types of relationships based on the WordNet database. The relationships are: extra-strong, strong, and medium strong. The deficiency of this approach as reported was in selection of lexical chains because (1) certain words were not included in chains in which they clearly belonged and (2) certain words were included in chains in which they did not belong. So there was chains misclassification and this can lead to information misrepresentation.

In [2], WordNet was used in the implementation dealing with some of the limitations in Hirst and St-Onge's algorithm [9]. They used the Brill's part- of-speech tagger [5] to aid in the selection of candidate words. They were able to prune the formation of lexical chain to prevent exponential growth of memory usage. The strength of chains were computed using it's length and homogeneity index, after which strong chains are identified and sentences extracted using one of three heuristics presented in [2]. All these lexical chain methods are based on the premise that related words tends to co-occur within a discourse unit of text, and cohesion is one of the surface indicators of discourse structures, which can be identified with lexical chains.

In this paper, we present a new algorithm for computing lexical chains based on only strong and direct relationships so as to reduce ambiguity to the minimum. Another algorithm for computing the strength of lexical chains based on our approach of building it is presented. In addition, we describe the algorithm used in extracting sentences from strong chains so as to increase the chances for producing a coherent and informative summary as possible with about 20%-30% of the original text. All of these have been packaged as software component tool to support content summarization. The tool is equipped with an interface to learn new words so as to promote its adaptability and enhance overall performance.

3. TheMethodology

Summarization has been described as a two step process [10]: (1) Building a source representation from source text and (2) Summary generation. In this section, we present a method for building lexical chains as the source representation using WordNet and a local word base as the knowledge source. The algorithm for extracting sentences to produce a coherent and informative summary is as well developed.

3.1 WordNet

WordNet is a lexical knowledge base developed at Princeton University. It consists of four data files for nouns, verbs, adverbs and adjectives, which are organized into synonym sets called synset, each representing one underlying

lexical concept. Its organization is based on semantic relationships. Some of these relationships are:

a. Synonym : words that can be substituted for each other
b. Antonym : words that are opposite in meaning
c. Hypernym : consists of the superset of a word
d. Hyponym : consists of the subsets of a word
e. Meronym : consists of the parts of a word
f. Holonym : consists of the containers of a word

A word may have more than one synset, each corresponding to a different sense of the word. An index file is associated with each data file and a list of pointers is attached to each synset. These pointers express relations between synsets. An example of semantic representations that can be derived from WordNet is:

{Automobile, machine, person, organism}

3.1.1 Word-base

The schema of our text file used as local word base is given as follows:

Word_Base (Word_code (string), Word (string), Part_of_speech(string), Word_sense (string))

The file allows for multiple instances of word senses to be stored in the file. This includes proper nouns and compound nouns.

3.2 Building Chains

The concept of lexical chains as a source representation is based on the fact that if a text is cohesive and coherent, successive sentences are likely to refer to concepts that were previously mentioned and to other concepts that are related to them [8]. Given an input text, each word is analyzed one at a time in an attempt to relate it with a word that has occurred in a previous sentence using one of the relationships presented by WordNet. A sequence of these words in which a relationship has being established is what is referred to as a chain. As relationships are found, the current word is being linked to the chain in which a relationship is found and when no relationship is found the word begins a new sequence of words forming a new chain. At the end of this process, the chains built can give us an ideal source representation for the text being analyzed.

3.2.1 Candidate Words

Our selection of candidate words is derived from St-Onge's method [9] of validation based on the WordNet database, which considered only nouns as indicated by WordNet and were validated using the following: (1) it must not appear in the stop-word list where the stop-word list contains closed-class words, and many vague high-frequency words that tend to weaken chains and (2) it must be in the WordNet noun base as is, or morphologically transformed as a noun.

We do not consider words that are less than or equal to two letters in length. This is to eliminate word that play an important role in the construction of most texts, but will be largely misinterpreted by WordNet. Examples are:

1. 'be' being misinterpreted as Beryllium when in actual sense the texts mean '*be*' as in 'you are to *be* there'.
2. 'a' being misinterpreted as 'angstrom unit'
3. 'an' misinterpreted as 'Associate in Nursing'

The cost of the exclusion of these words is preferable to the cost of their misinterpretation, which cannot be avoided if they were allowed. An exclusion-list of words is also maintained similar to

the stop-word list used by St-Onge. Our list consists of varying length words that are also misinterpreted by WordNet. Morphological analysis is also carried out to ensure words are compatible with its representation in the database. Also, in order to improve the accuracy of the selection of candidate words, we have used the Brill's-part-of-speech tagger [5] to identify nouns and the local word base to classify the proper nouns and compound nouns that are not available in WordNet

3.2.2 Chain Construction

With all the candidate words from a given text obtained, we proceed to the process of actually constructing chains from these words. Unlike all other previous methods for building lexical chains, we consider a more strict set of relationships between words. Since our aim is to create an informative as well as coherent summary. Informative in the sense that the information being communicated by the summary is the same as that of the intent of the author or as indicated by the structure of the text. The summary must emphasize on the subject the original text is communicating. The relationships we consider are

1. The repetition of a word or a synonym of it, as in the following examples.

   Car → Automobile
   Computer → Machine
   Book → Publication

2. Any relation indicated by a direct relationship of any other of the following WordNet relation: Antonym, Hypernym, Hyponym, Meronym, and Holonym. Like:

   Car → Truck
   Car → Roof
   Memory → Computer
   Key → Piano

These are the only relationships considered, motivated by the result that shows that they are easily identifiable human readable relationships that contribute more to ensuring the coherence of a given text. This is no surprise due to the fact that these texts are mainly developed for humans, if not only. And also, strongly related sentences extracted from a given text, tends to be as coherent as the original text when arranged in the order of appearance in the original text. This algorithm eliminates any deviations by the author from the main subject of the text and enables the use of less complex methods for computing chain scores. In this algorithm, relationships between words are not scored since they are all considered to be strong relations.

Each candidate word in a given text linked to an existing chain if it is validated by any of these relationships and if it cannot be validated with any existing chain, a new chain is started. At the end of this process, the number of chains produced is slightly higher than those produced by existing methods and deviations by the author are clearly indicated mostly by chains of length one and two where the length of a chain is the number of words that could be successfully linked to it. Also, no attempt is made to chain a word read from the given text to a word in an existing chain, which is separated from it with more than five sentences. This has also been shown to improve the coherence of the chain.

3.2.2 Scoring Chains

Before trying to make use of the chains produced, we first score the chains to determine which chains is useful using a formula based on the following expectation.

1. Given two chains of the same length, the chain with the higher number of repeated words is scored higher.

2. Longer chains are scored higher than shorter ones

In fulfilling these requirements we derived a formula as follows

Score (chain) = length (number of distinct words / length)

Chains with a single element immediately score zero, and other extremely weak chains are easily identified. These chains are removed after which the average score of the remaining chains are computed and chains whose score fall below the average are further eliminated. The resulting group of chains left is chains that do well in significantly representing the main discourse of the text and forms the basis for the extraction of sentences to create a summary.

3.2.3 Summary Extraction

In [2], three heuristics for extraction of sentences from strong chains were considered. They include

1. For each chain in the summary representation, choosing the sentence that contains the first appearance of a chain member in the text.
2. For each chain in a summary representation, choose the sentence that contains the first appearance of a representative chain member in the text.
3. For each chain, find the text unit where the chain is highly concentrated. Extract the sentence with the first chain appearance in this central unit.

Of all, they stated that the third heuristic produced the worst result being preceded by the first and second. We thereby extend the first and second heuristic in defining our algorithm for sentence extraction based on the source representation.

On analysis of a set of sample texts, we found that the transition from the main discourse of a text to a closely related subject either in the form of emphasis, explanation, or just total deviation can be accomplished in a single sentence found in the text. This observation also serves as basis for the sentence extraction algorithm. Sentences are hereby extracted from the chains as follows.

Given a list of chains,

1. For the first chain, we extracted the first sentence represented by a word in the chain
2. For the subsequent chains, we extract the first sentence that is represented by a word in the chain that is also represented by a word in the previous chain. If no such sentence exists, we extract the first sentence represented by a word in the current chain.

   The concept of a representative chain member is not considered here because we are trying to build a coherent summary as possible which would demand considering the best next sentence related to the previous as much as possible. And also, it produces a poor result as reported in [2].
3. And for the last chain in the list, besides extracting the sentence that overlaps with the previous, the last sentence represented in the chain is also extracted.

3.2.4 Packaging the tool

The various algorithms described so far, have been implemented and packaged as an ActiveX (COM) DLL (Dynamic Link Library) using the Activex Template Library (ATL) of Visual C++. It is therefore, an in-process server that can be invoked at runtime by various programming clients based on the signature of its IDispatch interface which defines the basis for connection. Also, we implemented a GUI interface from which the text summarizer functionality can be accessed by desktop users (See Figure 3)

The interface specification of the component as generated from the .idl file written in Microsoft Interface Definition Language (MIDL) dialect is shown in Figure 1.

```
import "oaidl.idl";
import "ocidl.idl";
        [
                Object,
                Uuid(13007041-BEC7-4D18-90F0-
                7BD83C090725),
                dual,
                helpstring("IContSumarizer
                Interface"),
                pointer_default(unique)
        ]
        interface IContSumarizer : IDispatch
        {
                [id(1), helpstring("method
                Sumarise")] HRESULT
                Sumarise([in] BSTR filepath, [out,
                retval] BSTR summaryfile);
        };
        Uuid(2FD42A78-F8BA-4498-9BE6-
        38ECC76C1756),
        version(1.0),
        helpstring("TextSummarizer 1.0 Type
        Library")
]
library TEXTSUMMARIZERLib
{
        importlib("stdole32.tlb");
        importlib("stdole2.tlb");
        [
                Uuid(1303F250-0BEA-4BC3-
                BAED-369FB068B38F),
                helpstring("ContSumarizer Class")
        ]
        coclass Sumarizer
        {
                [default] interface IContSumarizer;
        };
};
```

Figure 1. IDL Specification of the Content Summarizer DLL

The IDispatch interface (see highlight) specification shows that the functionality of the tool can be invoked by calling the method "Sumarise" with a single input string argument. The argument is the pathname of the document file to be summarized, while the method returns the name of the summary output file. A sample code for a typical instance of invocation of the DLL component from Visual Basic is given as follows:

```
Private Sub Summary_Click()
Dim filepathname, sumarryfilename as string
Filepathname = "c:\mydocuments\mypaper.txt"
smmaryfilename =
TextSummarizer.sumarise(filepathname)
debug.print summaryfilename
End Sub
```

Also, a typical instance of web scripting with the text summarizer component is given as:

```
<HTML>
<HEAD>
<TITLE>ATL sample code for object
ContSummariser </TITLE>
</HEAD>
<BODY>
<OBJECT ID="TextSummarizer"
CLASSID="CLSID: 2FD42A78-F8BA-
4498-9BE6-38ECC76C1756>
</OBJECT>
<SCRIPT LANGUAGE="VBScript"
<!--
Sub summitutton_Click()
  Dim summarystring as string
  Summarystring = TextSummarizer.summarise (ststring)
  ' Where ststring is text received from client
  Response.Write summarystring
End Sub
-->
</SCRIPT>
</BODY>
```

Figure 2 is a snapshot of the graphic user interface of the desktop version of the tool, which can be accessed as a full Windows-based application. It also permits the introduction of new words and usage senses into the knowledge base.

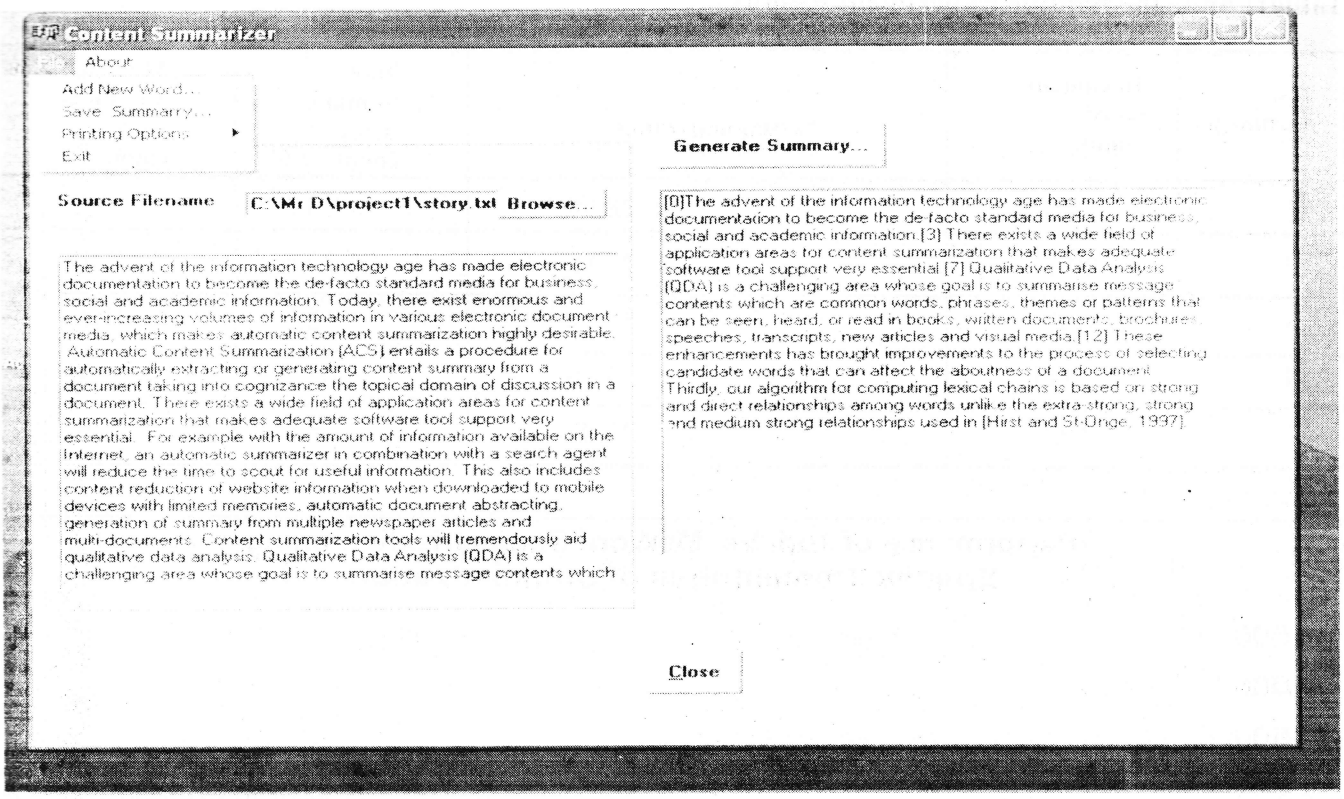

Figure 2. A Snapshot of the GUI of the Desktop Tool.

4. Evaluation

We assessed the tool by comparing its summaries with those obtained from language experts. Although, some of the summaries obtained from the experts were based on natural language understanding and text abstraction, the results from the tool based on text extraction proved satisfactory with a summary of about 20%-30% of the sentences in the original text. For example the result obtained in the summarization of the abstract of this paper can be seen as shown in Figure 2. We also compared the performance of our tool with the summarizer of Microsoft Word and obtained an appreciably good result. For example figure 3 shows the trend of closeness of performance of our tool summarizer when compared to the Microsoft Word Auto-summarizer in terms of word count of generated summaries. The procedure employed was to compute the percentage of original text in the summary generated by the tool summarizer and correspondingly generated a summary with the MS Word Auto-summarizer at the same set percentage (since MS summarizer can be configured to generate summaries at specific percentages). The word count obtained in the series of test procedures carried out shows close performance behaviour between the tool summarizer and MS word Auto-summarizer, although with the MS word summarizer having fewer words in most cases. A table showing some of the test results is shown in Table 1.

Table1: Showing Sample Experiment Results

Test documents	Document word count	% Summarization	Tool Summary word count	MS word Summary word count
t1	472	28.39	134	127
t2	881	25.88	228	214
t3	602	20.26	122	122
t4	2711	24.08	653	636
t5	1927	25.01	482	474

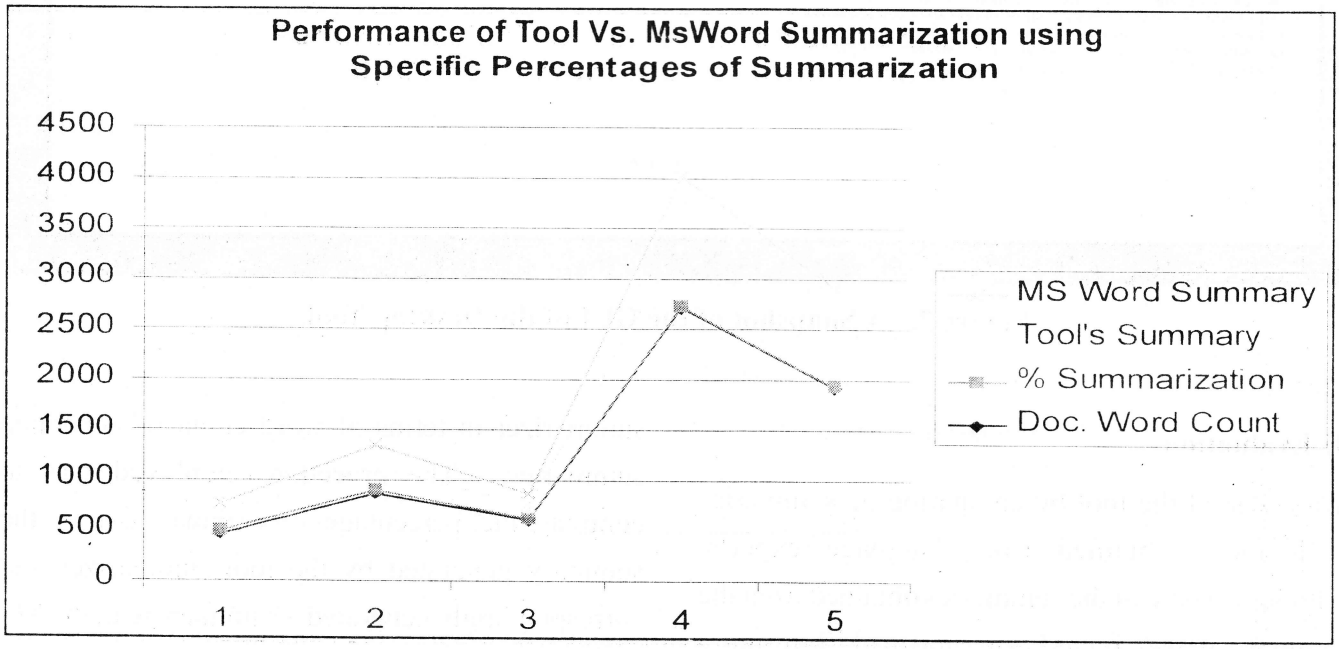

Figure 3. Performance of Tool Summarizer vs MS Word Summarizer

5. Conclusion

Although the length of summary produced this tool ranks favourably with the result produced by other algorithms, it performance also tends to result in a very coherent summary of the sample texts. The length of the summary may be further controlled by increasing or decreasing the boundary value between a strong and a weak chain in the algorithm used.

In future work we believe a better improved summary could be obtained by an algorithm that integrates into it, some method of parsing the text. This would give more control over the summary and may produce more specialized sentences that draw ideas over a range of sentences. A sentence reduction algorithm could also be used as this would further eliminate extraneous words from sentences.

However, the implementation of this tool as a plug-in component based on improved lexical chain algorithm that can be utilized by various clients applications and the encouraging results obtained is no doubt a step forward for context summarization.

References:

[1] Barzilay, R. (1997): Lexical Chains for Summarization. Master's Thesis, Ben-Gurion University, Beer-sheva, Israel.

[2] Barzilay, R. and Elhadad, M. (1999): Using Lexical Chains for Text Summarization. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*, pages 111-121. The MIT Press.

[3] Baxendale. P.ÊB. (1958): Man-Made Index for Technical Literature - an Experiment. *IBM Journal of Research and Development*, 2(4):354-361.

[4] Boguraev, B., and Kennedy, C. (1997): Salience based content characterisation of text documents. In ACL/EACL-97 Workshop on Intelligent Scalable Text Summarization, 29. Madrid: Association for Computational Linguistics and the European Chapter of the Association for Computational Linguistics.

[5] Brill, E. (1992): A simple rule-based part-of-speech tagger. In Proceedings of the Third Conference on Applied Computational Linguistics. Trento, Italy: Association for Computational Linguistics.

[6] Church, K.(1988): A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In Proceedings of the Second Conference on Applied Natural Language Processing, ACL, 136-143.

[7] Edmundson H.P. (1969): New Methods in Automatic Extracting. *Journal of the Association for Computing Machinery*, 16(2):264-285.

[8] Halliday, M. and Hassan, R., (1976): Cohesion in English, London: Longman.

[9] Hirst, G., and St-Onge, D. 1998 [to appear]. Lexical chains as representation of context for the detection and correction of malapropisms. In Fellbaum, C., ed., WordNet: An Electronic Lexical Database and Some of its Applications. Cambridge, MA: The MIT Press.

[10] Jones, K.S. (1993): What might be in a Summary? In Knorz, G.; Krause, J.; and Wornser-Hacker, C, eds. Information Retrieval '93: von der modellierung zur anwendung Konstanz, Universitatverlag Konstanz 9-26.

[11] Kupiec, J.; Pedersen, J.; and Chen, F. 1995. A trainable document summarizer. In Proceedings, 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 68-73. Seattle, Washington: Special Interest Group on Information Retrieval.

[12] Lin, C. and Hovy, E. (1997): Identifying Topics by Position. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 283-290. Association for Computational Linguistics.

[13] Luhn, H., P. (1958): The Automatic Creation of Literature Abstracts. *IBM Journal of Research Development*, 2(2):159-165, 1958.

[14] Mann, B. (1999): An Introduction to rhetorical structure theory.

[15] Mann, W., and Thompson, S. (1988): Rhetorical structure theory: Toward a functional theory of organization. Text 8(3):243281.

[16] Marcu, D. (1997): From Discourse Structures to Text Summaries. In Inderjeet Mani and MarkÊT. Maybury, editors, *Proceedings of the Workshop on Intelligent Scalable Text Summarization at the 35th Meeting of the Association for Computational Linguistics, and the 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 82-88, Madrid, Spain.

[17] Morris, J., and Hirst, G. (1991): Lexical cohesion computed by thesaural relations as an indicator of thestructure of text. Computational Linguistics 17(1):2143.

[18] Ono, K., Sumita, K. and Miike, S. (1994): Abstract Generation Based on Rhetorical Structure Extraction. In *Proceedings of the 15'h International Conference on Computational Linguistics*, pages 344-348, Kyoto, Japan

[19] Silber and McCoy. (2000): Efficient Text Summarization Using Lexical Chains. In *Proceedings of the ACM Conference on Intelligent User Interfaces (IUI'2000)*, January 9-12 2000.