

CERTIFICATION OF APPROVAL

Text Summarization

By

Siti Noor Arfah Binti Umar

Dissertation Submitted to the Information Technology Programme

Universiti Teknologi PETRONAS

In partial fulfillment of the requirement for the

Bachelor of Technology (Hons)

(Information Communication Technology)

Approved By,



.....
(Ms. Vivian Yong Suet Peng)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

June 2006

t

p

98.5

.A87

S622

1999

- 1) Automatic abstractingⁱ
- 2) Computational linguistics.

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein has not been undertaken or done by unspecified sources or persons.



.....
SITI NOOR ARFAH BINTI UMAR

ABSTRACT

Text summarization is the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks) [2]. By providing a text summarization system that will simplify the bulk of information and producing only the most important points, the task of reading and understanding a text would inevitably be made easier and faster. With a large volume of text documents, a summary of each document greatly facilitates the task of finding the desired documents and the desired data from the documents. As a solution for the above matter, this project objective is to simplify the texts from a previous text summarization system and further reducing the number of words in a sentence, shortening the sentences and eliminating sentences with similar meanings and also produce grammar rules that generate sentences that are human-like. The waterfall model is chosen as the project development life cycle. A detailed research has been conducted during the requirement definition phase and the system prototype is designed in the system and software design phase. During the development phase, the coding implementation will be conducted and the unit testing part will be done throughout that development process. After the entire unit has been tested, they will be integrated together and the system testing can be done as a whole. The complete program is put through thorough test and evaluation to ensure its functionality and efficiency. As the conclusion, this project should be able to produce a summarized text as the output product and meet the project requirements and objectives.

ACKNOWLEDGEMENT

Assalamualaikum and Alhamdulillah. First and foremost the author would like to take this opportunity to convey her gratitude and appreciation to Ms. Vivian Yong Suet Peng, her final year project supervisor who had supervised her progress throughout the development of the project. Her advice, comments and ideas has been the main factor for this project's success.

The author would also like to thank her family for all their advice and support through all the times the author was in need. The author would also like to acknowledge her friends namely Mohd Fakri Bahri, Sarah Haryati Zulkifli, Nurunnisa Abd Aziz and all the others who has assisted and helped in anyway in making this project a success.

A well-known Malay proverb translated to "*One can pay back the loan of gold, but one dies forever in debt to those who has given kind assistance.*" As such, the author is forever in debt of those who has helped her in any way. Thank you

CHAPTER 4:	RESULT AND DISCUSSION	. . .	16-22
	4.1 Functional Testing	16
	4.2 Integration Testing	17
	4.3 Data Gathering and Analysis.	18-21
	4.3.1 Sample of Input data.	19
	4.3.2 Sample Result and Analysis	19-20
	4.3.3 Evaluation Result	20-21
	4.4 Discussion	21-22
	4.5 System Limitations	22
CHAPTER 5:	CONCLUSION AND RECOMMENDATION.	. . .	23-24
	5.1 Conclusion	23-24
	5.2 Recommendation and Future Works.	24
REFERENCES	25-27

LIST OF FIGURES

Figure 3.1	The Waterfall Model	7
Figure 3.2	System Architecture	9
Figure 3.3	Text Summarization System Interface	10
Figure 3.4	Sample Output of Stemmer Module	11
Figure 3.5	Sample Output of Redundant Word Removal Module.	12
Figure 3.6	Sample Grammar Rules	13
Figure 3.7	Sample of Sentence Rule Application.	14
Figure 4.1	Three Main Buttons	16

LIST OF TABLES

Table 4.1	Integration Testing Comparison	18
Table 4.3.1	Result and Analysis for Text 1	19
Table 4.3.2	Result and Analysis for Text 2	20
Table 4.3.3	Result of Evaluation	20

CHAPTER 1

INTRODUCTION

1.1 Background of study

Automatic text summarization has received a great deal of attention in recent research. The rapid growth of the Internet has resulted in enormous amounts of information that has become increasingly more difficult to access efficiently. The ability to summarize information automatically and present results to the end user in a compressed, yet complete form would help to solve this problem [1].

Text summarization is the *process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks)* [2]. By providing a text summarization system that will simplify the bulk of information and producing only the most important points, the task of reading and understanding a text would inevitably be made easier and faster.

Automatic text summarizing is similar with both Information Retrieval and Extraction. Information Retrieval is basically retrieving information based on a certain criteria or set of words among a large amount of data while Information Extraction can be described as processing a document with the objective of revealing only the relevant information and creating a more concise document. Summarization that is done by methods of extraction processes documents as a collection of sentences. It identifies and returns only the sentences that are considered most relevant thus; the summary is a subset of the set of sentences of the original text. Summarization by abstraction, on the other hand, applies more

complex linguistic technology techniques that will generate the output by processing the information in the original text. This way of summarization is complex and needs an advanced understanding of the Natural Language Processing techniques of Artificial Intelligence.

This project will be focusing on the semantics-side of language processing. The main issue that would be touched on would be of Semantic Analysis that can be defined by assigning meaning representation to inputs. Summaries can be built on a deep semantic analysis of the source text. For example in 1995 a research was done to investigate ways to produce a coherent summary of several texts describing the same event, when a full semantic representation of the source texts is available. This type of source abstraction is the most expressive, but very domain dependent [3]. A great number of grammar rules need to be defined and semantics need to be elaborated upon in order to produce a satisfactory result.

1.2 Problem statement

In our current digital era, everything has been made simpler with the help of machines that enable us to perform difficult and tedious work with greater ease. But as there are more technology and tools, more time has to be spent in learning to understand and operate the technologies. This in turn will require humans to read in order to gain the information and knowledge required. Thus now, we end up with a lot of technology to learn about but with very limited time to read every single thing about it. This is where a text summarization system comes as a necessity in our life. With a large volume of text documents, presenting the user with a summary of each document greatly facilitates the task of finding the desired documents and the desired data from the documents.

The issue of text summarization has been brought up years ago and research on the topic has been done since many years back. Although it has been accepted as an important matter to work on, not many lights have been shed into this area. In Malaysia, this area of Artificial Intelligence has not been studied in depths into nor has it been given much attention.

A previous text summarization system [7] has been developed to extract important sentences from a given text. But since the sentence extracted could still be improved by applying techniques that concerns with the grammatical structure of the text, a more in-depth system could be developed to fine-tune the current system.

1.3 Objectives

1. To build a text summarization system that would summarize and simplify the texts from a previous text summarization system in order to further reduce the number of words in a sentence.
2. To research and try to find a way to develop a text summarization system that will summarize a text by shortening the sentences and eliminating sentences with similar meanings.
3. To develop a system that contains as many grammar rules as possible in order to generate sentences that is as human-like as possible.

1.4 Scope of study

This project will be concerned about producing a text summarization using semantics of the English grammar rules. The input for this project would be the output from the previous project [7] done on text summarization using a neural-based approach. The scope of this project is to simplify sentence structures and to eliminate same meanings in the text.

CHAPTER 2

LITERATURE REVIEW AND THEORY

2.1 History and previous works

The research of automated text summarization can be said to have started around the late fifties. Attempts to produce a human quality summary have shown that a text summarization system has to include understanding of the meaning of the sentences itself, abstraction and words production [4]. From the very beginning, the subsequent task of automatic abstracting has been considered a problem that can be solved using surface-level pattern matching techniques and domain-independent as well as language-independent statistical methods.

According to the level of semantic analysis, summarization methods can be roughly classified into the following 3 categories [14]:

- 1) **Based on extraction.** These methods analyze the sentences similarity and extract the most important sentences to form the summary. MEAD [13] is an example of this category. MEAD is the most elaborate publicly available platform for multi-lingual summarization and evaluation. MEAD implements a battery of summarization algorithms, including baselines (lead-based and random) as well as centroid-based and query-based methods. Its flexible architecture makes it possible to implement arbitrary algorithms in a standardized framework. [13]

- 2) **Based on simple semantic analysis** such as Lexical Chain. We first construct a tree structure of the origin document, and then score the every chain to select the strongest chains as output. The BioChain Project [15] propose concept chaining to link semantically-related concepts within biomedical text together. The resulting concept chains are then used to identify candidate sentences useful for extraction. The extracted sentences are used to produce a summary of the biomedical text [15].

- 3) **Based on deep semantic analysis.** For example, Marcu proposed an approach based on the construction of a rhetorical tree that uses explicit discourse markers and heuristic rules to decide which is the best rhetorical tree for a given document [16].

However, most of the automatic summarization will in the end boil down to a sentence extraction problem. Summarization systems can either extract text-spans related to the main *topics* of a whole document or apply a query-based summarization that will produce abstract information relevant to a given query. Many numbers of works could be found for researches done to produce an automatic text summarization. Microsoft Word has had a summarizer for documents since 1997. R. Barzilay and M. Elhadad developed a method that creates text summaries by finding lexical chains from the document [3]. This project applied the “simple semantic analysis” as discussed above. This project however differs from those of BioChain’s as it discusses a more general text for summarization rather than scope down to a narrow field.

B. Hachey and C. [8] Grover presented favorable sentence extraction results in classification and ranking frameworks. By applying a breakdown of sentence extraction scores by rhetorical category they have reported that rhetorical information is an important means of controlling argumentative distribution of sentences in an extractive summarization system [8]. Next is the SUMMARIST text summarizer from the University of Southern California that also applies a sentence extraction method for summarization. It is a system that combines symbolic concept-level world knowledge with robust NLP processing to overcome the

problems of the depth/robustness tradeoff strives to create text summaries based on the equation: summarization = topic identification + interpretation + generation [9].

Meanwhile summarizations in other languages have also been produced such as for Turkish, German, Norwegian and many more others. SweSum is the first automatic text summarizer for Swedish based on statistical, linguistically and heuristic methods where the summarization system calculates how often certain key words appear [10]. The summarization system calculates the frequency of the key words in the text, which sentences they are present in, and where these sentences are in the text. It considers if the text is tagged with bold text tag, first paragraph tag or numerical values. All this information is compiled and used to summarize the original text.

From the point of view of natural language processing, producing a semantically related text summarization is considered a heavily knowledge-based task requiring a substantial knowledge background [11]. Designing computer systems to understand natural language input is a difficult task in order to produce a human-like summary. The very intensive and complex computational grammars behind natural language applications are often inefficient, incomplete and ambiguous [12]. This obvious difficulty in constructing adequate grammars has motivated much research in machine learning.

Semantic grammars, which uniformly incorporate both syntactic and semantic constraints to parse sentences and produce semantic analyses, have proven extremely useful in constructing natural language interfaces for limited domains [4]. But still, interpreting a sentence is not a simple thing to describe much less to produce a summary of it. Theories for computer processing of natural language will often insist on the necessity of a world representation for the interpretation of a sentence or a set of sentences. But it is maintained that a semantic grammar should mainly include a clear relation, not only between an expression of a language and the objects to which they refer in a particular usage, but also between the sense of the expression and their references.[5]

CHAPTER 3

METHODOLOGY

3.0 Software Process Model

The methodology applied in this project is the most common of life cycle models namely the Waterfall Model. This type of software development methodology was selected mainly because it is very straightforward and easy to understand and uncomplicated to perform. In a waterfall model, each phase must be completed in its entirety before the next phase can begin as shown in Figure 1. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project [19].

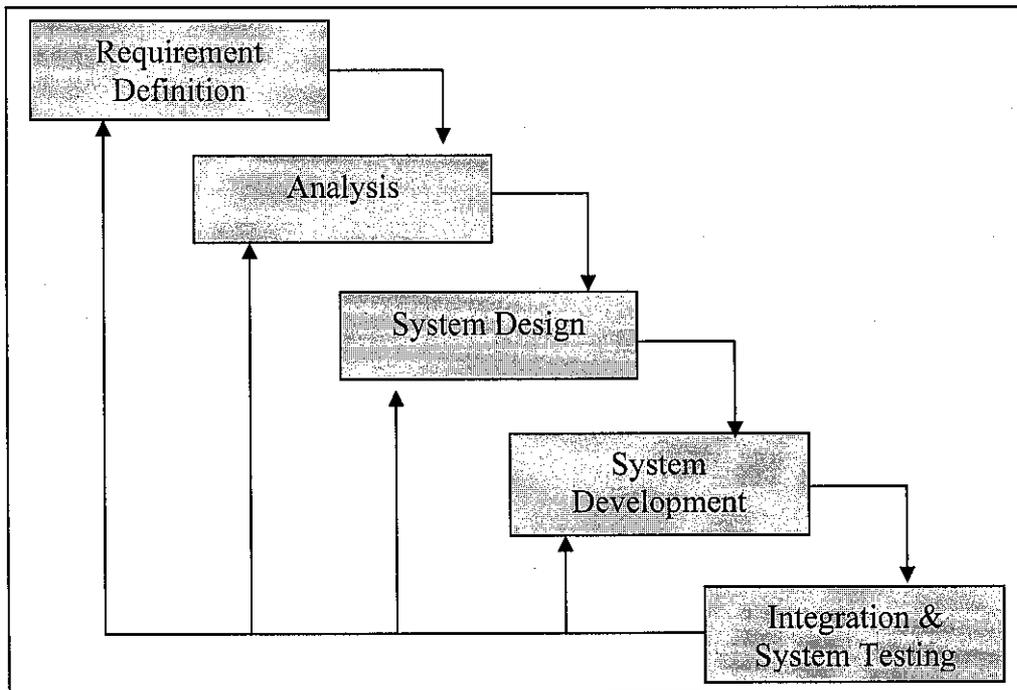


Figure 3.1: The Waterfall Model

3.1 Requirement Definition

Requirements are set of functionalities and constraints that the end-user (who will be using the system) expects from the system [18]. The requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is also studied. All the requirements will in the end set the constraints for the system and the functions that the system will need to incorporate.

3.2 System Design

The design phase is important in order to understand what is going to be created and what it should look like. The requirement specifications from first phase are studied in this phase and system design is prepared [18]. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture [18].

The flow of the system is that the user will have to provide an input and the system will produce a summary of the input text. Figure 3.2 below illustrates the system architecture of the summarization system.

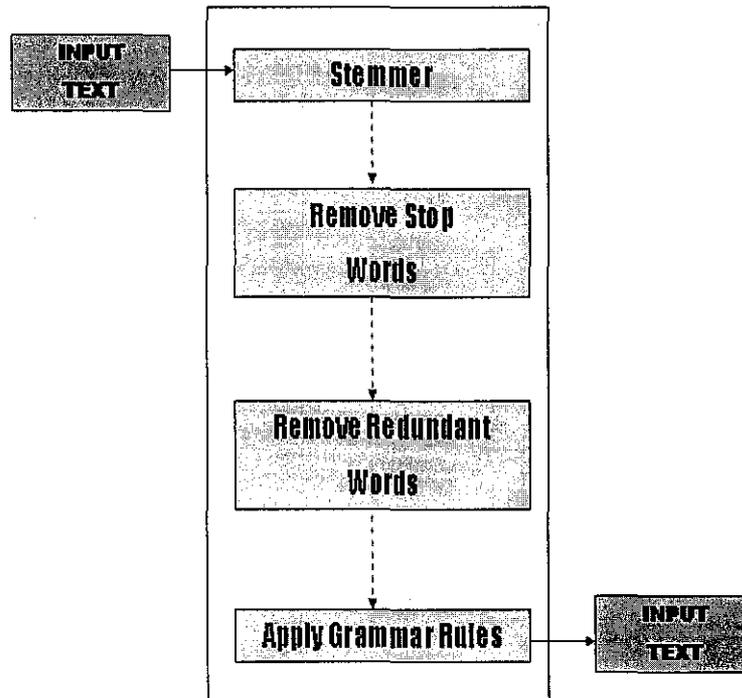


Figure 3.2: System Architecture for the Text Summarization System

From the figure, the system components can be divided into 4 parts. They consist of a stemmer, stop words remover, redundant words remover and lastly application of the grammar rules. The stemmer will remove suffix and prefixes in each word. For example the word transitional will be stripped of its suffix “ional” and the output will be transit. The stemmer is important to ensure that the words that are actually the same but has suffix and prefixes will be identified and recognized as the same word instead of words with different meanings. The next stage or module is to remove the stop words. Stop words are those words which are so common that they are useless to index or use in searches and are thus removed to simplify a sentence and to leave only words with significant meanings. Example of stop words includes words such as “because”, “although”, “don’t” and lots more. Next process is to remove redundant words. This is done by applying a ranking system that will detect the words in a sentence that have the same meaning and will select only two words to be displayed as the output randomly. The ranking algorithm was constructed purposely for this system to select which word to be selected out of the list of words with the same meaning. Lastly the sentence that has been stripped of unnecessary words and elements will be applied the grammar rules that will restructure the sentence to produce a summary of the text.

In this phase the interface for the system was designed. Figure 3.3 shows the interface design for the text summarization system. The functions that are available in the user interface consist of the “summarize”, “clear” and “exit” button. The user will open the text file they want to summarize by clicking on the “browse” button and it will be immediately be displayed on the text field above it. The user will then have to click the “summarize” button for the text to be run through the system and the output to be displayed in the text field below. The “clear” button will remove the current text in the fields and thus clear all the words in the field. Clicking the “exit” button will terminate the program.

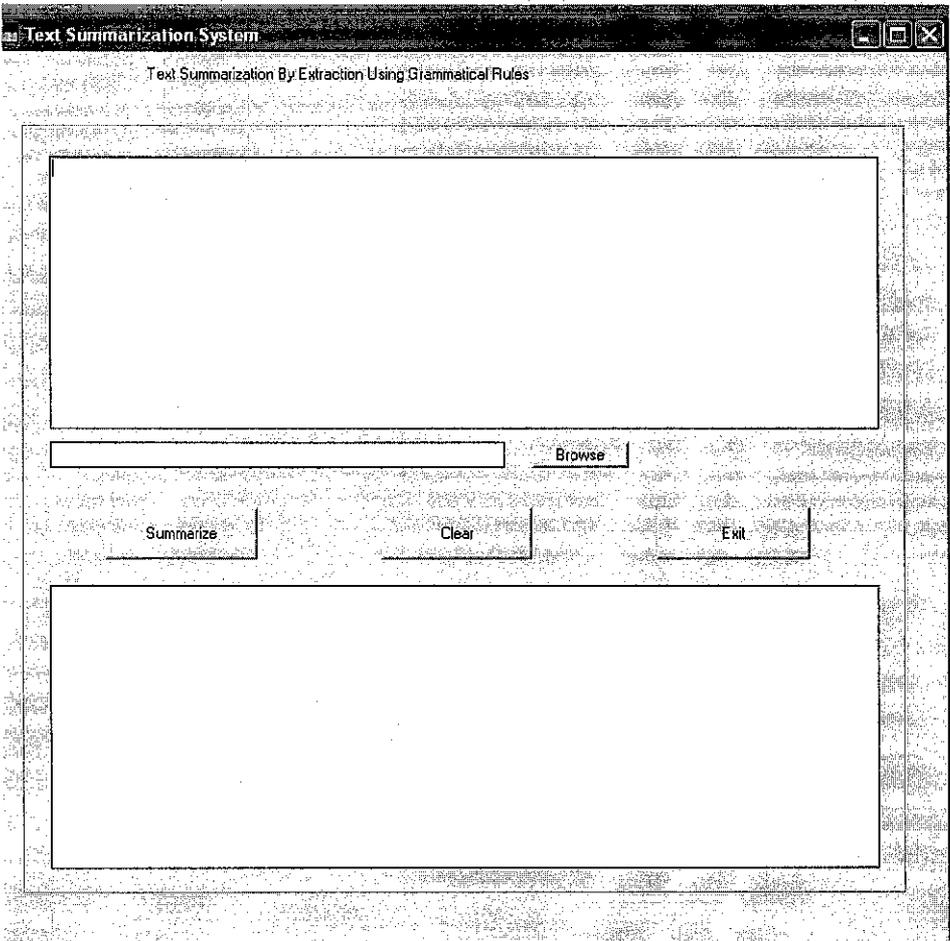


Figure 3.3: Text Summarization System Interface

3.3 System Development

Once the system design phase is completed, the work was divided into modules/units and construction began. The system is first developed in small programs called unit which was developed and tested for its functionality. Testing each module separately to ensure it is working is referred to as Unit Testing. Among the module that was constructed for this system was the stemming module, stop words removal module, ranking module and the module in which the grammar rules are applied to the text.

3.3.1 Stemmer

The stemmer used in this system is an altered code adapted from a stemmer available on the internet. The Porter stemming algorithm (or 'Porter stemmer') is a process for removing the commoner morphological and inflexional endings from words in English. Its main use is as part of a term normalization process that is usually done when setting up Information Retrieval systems. Examples of how the system input and output would look like are as in Figure 3.4:

```
| ?- stem_token(w([r,a,i,d,e,d]), P).  
P = w([r,a,i,d])  
  
| ?- stem_token(w([r,a,i,d,e,r]), P).  
P = w([r,a,i,d,e,r])  
  
| ?- stem_token(w([r,a,i,d,e,r,s]), P).  
P = w([r,a,i,d,e,r])  
  
| ?- stem_token(w([l,o,o,k,i,n,g]), P).  
P = w([l,o,o,k])
```

Figure 3.4 : Sample output from stemmer module

3.3.2 Stop words remover

The stop word remover module is (as the name implies) to remove all the stop words in the text. More than 850 stop words have been identified and included in the stop word “database” for this system.

3.3.3 Redundant words remover

This module will look through the sentence and identify the words that occur more than three times and will not allow any words to be repeated more than three times in the line. This is just the initial function of this module as the real module would identify each word in a category. Each category of words consists of words with the same meaning as they occur in the thesaurus. This is done by applying a ranking system that will detect the words in a sentence that have the same meaning and will select only two words to be displayed as the output randomly. The ranking algorithm was constructed purposely for this system to select which word to be selected out of the list of words with the same meaning. The module functions to remove words in the same category that occur more than 3 times in one sentence. Figure 3.5 is an example depicting how the input and output of the redundant word remover currently works.

```
| ?- remove_triplets( [1,3,4,4,4,2,2,1,5,5,5,5,5,1,6], P ).  
P = [1,3,4,2,2,5,6]  
  
| ?- remove_triplets( [pretty, beautiful, cute, cute, pretty, cute, gorgeous], P ).  
P = [pretty, beautiful, cute, pretty, gorgeous]
```

Figure 3.5: Sample output from redundant word removal module

3.3.4 Application of grammar rules

The grammar rules applied in this system are based on semantic rules applied in most linguistic systems. A sentence is made up of different combination of sentence structures such as the determiner, nouns, verbs, adjectives, conjunctions and other grammar structures. Semantic rules is concerned with providing the system with as much sentence structure as possible to cater for all the sentences that might be used as the input for the system. As it is impossible to cater for all the sentences in the English language, this system is catered to suit texts in the fields of Information Technology only. Figure 3.6 show examples of the grammar rules that have been currently developed for the system.

```
%English grammar Rules

a(z,a(DET,N)) --> det(z,DET), n(z,N).
b(z,b(IS,VED))-->si(z,IS), ved(z,VED).
c(z,c(ST,DET))-->st(z,ST), det(z,DET).
d(z,d(V,ST))-->v(z,V), st(ST).
e(z,e(FIELD,MED))-->field(z,FIELD),
med(z,MED).
f(z,f(PREPT,ST))-->prept(z,PREPT),
st(z,ST).
g(z,g(ST,ST))-->st(z,ST), st(z,ST).
h(z,h(MED,ADV))-->med(z,MED), adv(z,ADV).
i(z,i(ST,DET))-->st(z,ST), det(z,DET).
j(z,j(FIELD))-->field(z,FIELD).
k(z,k(VED))-->ved(z,VED).
l(z,l(A,ST))-->a(z,A), st(z,ST).
m(z,m(J,B))-->j(z,J), b(z,B).
n(z,n(C,D))-->c(z,C), d(z,D).
o(z,o(E,F))-->e(z,E), f(z,F).
p(z,p(H,J))-->h(z,H), j(z,J).
q(z,q(K,TIME))-->k(z,K), tme(z,TIME).
r(z,r(J,Q))-->j(z,J), q(z,Q).
v(z,v(M,N,O))-->m(z,M), n(z,N), o(z,O).
t(z,t(K,P,R))-->k(z,K), p(z,P), r(z,R).
u(z,u(L))-->l(z,L).
s(z,s(U,V,T))-->u(z,U), v(z,V), t(z,T).
```

Figure 3.6: Sample Grammar Rules

3.5 Tools Required

3.2.1 Hardware

- Personal Computer AMD Athlon 64 Processor 2.0Ghz.
- Memory space 256MB RAM.

3.2.2 Software

- Platform Microsoft Windows XPsp2.
- LPA WIN-Prolog 4320

CHAPTER 4

RESULT AND DISCUSSION

4.1 Functional Testing

Once the system is completed it is put through an assessment to test each function offered as the interface, to ensure that it was working faultlessly. As there are three main buttons with different functions that the end user would deal with, each would be tested to ensure functionality.



Figure 4.1: The three buttons with main functions

Button	Expected Result	Actual Result	Remarks
“SUMMARIZE”	Summarize the given text input and produce a summarized text as output	The text input was summarized and the output was a shortened sentence	BUTTON IS FUNCTIONAL
“CLEAR”	To clear all fields and reset each functions	All fields were cleared and each functions reset	BUTTON IS FUNCTIONAL

"EXIT"	Exit from the system and close the current window	The system was successfully exited and the window closed	BUTTON IS FUNCTIONAL
--------	---	--	----------------------

Table 4.1: Functional testing result

4.2 Integration Testing

The code produced as the engine for the system will be integrated and tested to verify its functionality. The mechanism or the coding will be tested on its own without the interface to make sure that it functions to meet the objectives set. The system will also be tested once it is linked with the interface to ensure that it functions as well as it did without the interface. The test on the system that has been integrated with the user interface is also crucial to ensure that all the linkage was properly done and that the user will be able to manipulate it as required.

Without Interface	With Interface
<p>The functionality is tested by invoking the system and accessing it through the Win-Prolog console manually. The system functions are tested using the prolog programming language.</p>	<p>After making sure that the program is functioning correctly, the "code" is linked to the user interface which is where the end user will be accessing the system. This time the system is tested using the buttons in the interface to manipulate the program.</p>

<p>The program works well and produced an acceptable output. The text sentences used as input were successfully summarized and a shorter result was produced.</p>	<p>The system functions correctly even when linked to the user interface. The buttons works and the user will not need to know any prolog programming in order to use the system.</p>
---	---

Table 4.2: integration testing comparison

4.3 Data Gathering and Analysis

Once the system has been successfully completed, an evaluation is carried out to test the efficiency of the system and its effectiveness in meeting the objectives set. The evaluation is carried out by first finding sample texts that consists of output from the previous system [7] or any text similar to it. These texts have already been processed to leave only the important sentences or paragraph instead of a large text file.

10 paragraphs from texts were selected randomly to be tested for this evaluation purpose. Each contains sentences that are very long due to the number of words in the sentence (more than 10 words in a given sentence). The number of words in the sentence that is used as the input is documented. After the sentence is run through the system, the output is also documented and the number of words after the test is noted. A comparison of the number of words reduced is performed and the result is documented.

4.3.1 Sample of input data

- Fingerprints have been routinely taken, categorized, and filed for over 100 years, and since the 1980s have been digitized, stored, shared, and compared on networked computer systems. Fingerprints are accepted by all courts worldwide as positive proof of identity, and a considerable body of knowledge has been established and is legally accepted regarding fingerprint identification methods.
- The tools for the search and seizure side of computer forensics are a sophisticated potpourri primarily focused on the physical side of computing: tracing and locating computer hardware, recovering hidden data from storage media, identifying and recovering hidden data, decrypting files, decompressing data, cracking passwords, crow- barring an operating system by bypassing normal security controls and permissions, and so forth.
- Although most research on visually induced motion sickness has been on sickness induced in vehicle simulators or simulator sickness, it is assumed that the problems and findings generalize to other virtual environments. Furthermore simulator sickness is a subset of the motion sickness experienced from travel through virtual environments, for which we suggest the more general term cybersickness.
- Grid computing offers a model for solving massive computational problems by making use of the unused resources such as CPU cycles or disk storage, of large numbers of disparate computers, often desktop computers, treated as a virtual cluster embedded in a distributed telecommunications infrastructure. Grids offer a way to solve Grand Challenge problems like protein folding, financial modeling, earthquake simulation and climate/weather modeling.

4.3.2 Sample result and analysis

Input Text	Fingerprints have been routinely taken, categorized, and filed for over 100 years, and since the 1980s have been digitized, stored, shared, and compared on networked computer systems. Fingerprints are accepted by all courts worldwide as positive proof of identity, and a considerable body of knowledge has been established and is legally accepted regarding fingerprint identification methods.
Output Text	Fingerprints have been taken, categorized, filed, digitized, stored, shared and compared on computer systems. Fingerprints are accepted by courts as proof of identity and the knowledge has been established and accepted regarding fingerprint identification methods.
Word Count	35 / 56

Table 4.3.1: Result and analysis for Text 1

Input Text	Grid computing offers a model for solving massive computational problems by making use of the unused resources such as CPU cycles or disk storage, of large numbers of disparate computers, often desktop computers, treated as a virtual cluster embedded in a distributed telecommunications infrastructure. Grids offer a way to solve Grand Challenge problems like protein folding, financial modeling, earthquake simulation and climate or weather modeling
Output Text	Grid computing offers a model for solving computational problems by using the unused resources such as CPU cycles or disk storage of large numbers of computers treated as a virtual cluster embedded in a distributed telecommunications infrastructure. Grids offer a way to solve problems like protein folding and financial modeling.
Word Count	50 / 65

Table 4.3.2: Result and analysis for Text 4

4.3.3 Evaluation result

Text	Words Before	Words After	Number reduced	Percentage of Reduction
Text 1	56	35	21	37.5%
Text 2	59	39	20	33.9%
Text 3	67	38	29	43.3%
Text 4	65	50	15	23.1%
Text 5	60	31	29	48.3%
Text 6	59	42	17	28.8%
Text 7	63	41	22	34.9%
Text 8	73	39	34	46.6%
Text 9	62	34	28	45.2%
Text 10	65	43	22	33.8%
Average percentage reduced				37.54%

Table 4.3.3: Result of Evaluation

After each text has been processed by the system the result is as shown in Table 4.3.3 above. The text with the most number of words is Text 8 while the least number of words occurred in the input text of Text 1. The average number of words put through the system for this evaluation purpose is 62.9 per paragraph. The output text with the most significant change was Text 5 with a result of 31/60 or with a 48.3% reduction from the original text. The number of words that the system manages to reduce varies from one text to another. This is because each sentence structure is different. Some might have more words with similar meanings or words from the adjectives category which will be eliminated under certain conditions. The average number of words the system was able to eradicate was 37.54% out of all the input texts. Although 37.54 is not such a big number, in text summarization it would be enough to simplify and reduce the number of words a person would have to read to understand a text. For example, if a text contained 1000 words that the user would have to read, by going through this summarization process the system could reduce an estimate of 375 words (based on the percentage of reduction obtained from the evaluation process).

4.4 Discussion

Based on the gathered data and the analysis done in the evaluation process, the system can be said to meet its objectives and is a solution that fits its problem statement. All the input sentences have successfully been shortened and the number of words in the text reduced, which is the most important thing required of a text summarization system.

This system is a semantic based summarization that is produced by applying predefined semantic grammar rules to the input text to produce a reduced and shortened output. As such, only the sentences with the sentence structure defined in the database will be recognized. Thus not all the texts and sentences will be able to be summarized by this system. This system is focused to the field of computers only as the database only contains grammar rules that are applicable to sentences with words

related to the field. Although semantic grammar is sometimes thought of as a very obscure system to produce, it have proven extremely useful in constructing natural language interfaces for limited domains as it incorporate both syntactic and semantic constraints.

When integrated with the previous text summarization system [7] the system could be used by anyone studying the field of Information Technology to simplify what they have to read. The previous system will identify and extract the sentences with important points while this program will process the extracted sentences to reduce the number of words and shorten the text.

4.5 System Limitations

The main limitation of the system is that it can summarize only one sentence at a time and thus cannot recognize more than one sentence if it is used as the input. Although the inputs are in the form of paragraphs, each sentence in the paragraph has to be fragmented into sentences and each sentence is summarized separately.

Using the current user interface, the system user cannot directly open the text file into the text field as the open file function is unavailable. This means that the user can only copy the sentences from another (previously) opened file and paste it onto the input field.

As there was a time limit to this project, only a limited amount of grammar rules can be produced as the engine of this system. Thus the system lacks grammar rules and can only summarize sentences whose grammar rules have been predefined in the database. Applying more grammar rules would make this system more versatile effective and efficient.

As the system uses the semantic grammar, it was only feasible to select one area of research in which the database would be concentrated on. The system could only summarize texts with topics and words related to the field of information technology or computers.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

Until today, numerous projects have been developed to produce a text summarization system that would efficiently minimize and reduce the words in a text. The most popular method of producing a text summarization is the lexical chain method and the statistical method. Even in UTP a text summarization system has been produced [7] a few years ago that applied the neural network in order to produce a concise and precise summary. As the text summarization system [7] managed to correctly and efficiently pick out the important points in a text, an enhancement is the only best thing to do to make a perfect text summarization system. This is a project that should fine-tune the result from that project and filter the sentences to make it even more concise.

By using the system, the sentences will be processed to produce a more simplified version of the output from the previous system [7]. A lot of people would benefit by using this system such as teachers, lecturer or student and even business-personnel and also technicians. Teachers or students would inevitable be reading a text to learn from its contents while a businessmen of technicians would read a text to understand and make decisions on what action to take based on a given text. In either case, a summarized text would help plenty.

The functionality of each module has been tested and the system was tested again upon integration. Since its completion the system has been able to constantly produce a shorter, more readable version of the text when the grammar rule is defined in the database. As a conclusion, this system has in deed met its objectives

by simplifying the texts from a previous text summarization system and further reducing the number of words in a sentence, shortening the sentences and eliminating sentences with similar meanings and also contains many grammar rules that generate sentences that are human-like.

5.2 Recommendation and Future Works

First and foremost, as this system is based on semantic grammar, the number of grammar rules produced would have to be countless in order to cater for all the sentence combination that could be produced in the English language. In addition to that, the system currently only caters for sentences in the field of Information Technology and Computers as the grammar rules were only designed for that specific field. Thus the number of grammar rules produced in this project is still not enough to produce an ideal text summarization system. Consequently in the future, more grammar rules should be produced and added to the database to make this system more effective and efficient.

The interface of this system could be reconstructed to include and allow more functionality to make it more user-friendly. Other functions that could be added include an automatic word count to show how many words have been produced and reduced by the program or enable the “open file from a specific location” function so that users wouldn’t need to copy and paste the input into the input field.

In the future, the system could also be integrated with other software to make it more attractive or to enhance the functionality of the system. For example, a method could be designed to enable the integration of Win-Prolog with Microsoft Access or Oracle to act as a database in which the data could be stored instead of the current “.pl” file. Java or Visual Basic could be used to make the user-interface more user-friendly with more functions. Another way is to integrate the prolog file with PHP or ASP to make it a web-based system so that it could be accessed through the internet and serve more people from all around the world.

REFERENCES

- [1] **“Efficient Text Summarization Using Lexical Chains”**
H. G. Silber and K. F. McCoy 2000. Retrieved 23 Dec 2005 from the World Wide Web : <<http://acl.ldc-upenn.edu/W/W00/W00-1438.pdf>>

- [2] I. Mani and M. T. Maybury. 1999 **“Advances in Automatic Text Summarization”**, The MIT Press, Massachusetts Institute of Technology, Cambridge, Massachusetts.

- [3] **“Using Lexical Chains for Text Summarization”**
R. Barzilay and M. Elhadad 1997. Retrieved 19 Dec 2005 from the World Wide Web: <<http://www-nlpir.nist.gov/projects/duc/pubs/2001papers/lenthbridge.pdf>>

- [4] **“Automatic Text Summarization Based on Word-Clusters and Ranking Algorithms”**
Massih R. Amini, Nicolas Usunier, and Patrick Gallinari 2005. Retrieved 19 Dec 2005 from the World Wide Web : <http://www-connex.lip6.fr/download_article/1031.pdf>

- [5] **“Learning Semantic Grammars with Constructive Inductive Logic Programming”**
J. M. Zelle and R. J. Mooney 1993. Retrieved 2 Feb 2006 from the World Wide Web:<www.cs.utexas.edu/~ml/papers/chill-aaai-93.pdf>

- [6] **“Semantic for Text Processing”**
Jean-Guy Meunier 1980. Proceedings of the 8th conference on Computational Linguistics . Retrieved 2 Feb 2006 from the World Wide Web: <<http://ucrel.lancs.ac.uk/acl/C/C80/C80-1054.pdf>>

- [7] Yie C. Y. 2004 “**A Text Summarization System With Neural-Based Approach**”, Universiti Teknologi PETRONAS.
- [8] B. Hachey and C. Grover (2005). “**Automatic Legal Text Summarisation: Experiments with Summary Structuring**”.
 Proceedings of the 10th International Conference on Artificial Intelligence and Law (ICAIL 2005), Bologna, Italy. Retrieved 12 Feb 2006 from the World Wide Web : <<http://homepages.inf.ed.ac.uk/bhachey/PUBS/icail05-author.pdf>>
- [9] **SUMMARIST Automated Text Summarization Website (2000)** Browsed on 19 Dec 2005 on the World Wide Web : <<http://www.isi.edu/natural-language/projects/SUMMARIST.html>>
- [10] **SweSum - A Text Summarizer for Swedish.**
 Dalianis H. 2000. Retrieved 2 Jan 2006 From The World Wide Web : <<http://www.nada.kth.se/~hercules/Textsumsummary.html>>
- [11] D. Radev, T. Allison, S. Blair-Goldensohn, J. Blitzer, A. Çelebi, S. Dimitrov, E. Drabek, A. Hakim, W. Lam, D. Liu, J. Otterbacher, H. Qi, H. Saggion, S. Teufel, M. Topper, A. Winkel and Z. Zhu.
“MEAD - a platform for multidocument multilingual text summarization.”
 In Proceedings of LREC 2004, Lisbon, Portugal, May 2004. Retrieved 12 Feb 2006 from the World Wide Web : <<http://tangra.si.umich.edu/~radev/papers/lrec-mead04.pdf>>
- [12] Q. Zhou, L. Sun, 2005, “**IS_SUM: A Multi-Document Summarizer Based on Document Index Graphic and Lexical Chains**”
 Chinese Academic of Sciences, J.-Y. Nie, Universite' de Montreal.
 Retrieved 12 Feb 2006 from the World Wide Web:
 <<http://www-nlpir.nist.gov/projects/duc/pubs/2005papers/cas.zhou.pdf>>
- [13] L. Reeve, H. Han, A. D. Brooks, 2005. “**BioChain: Lexical Chaining Methods for Biomedical Text Summarization**”
 Retrieved 12 Feb 2006 from the World Wide Web:
 <<http://www.pages.drexel.edu/~lhr24/pubs/2006SAC-BIO-130.pdf>>
- [14] Marcu, D. “**From Discourse Structures to Text Summaries**”.
 In The Proceedings of the ACL'97/EACL'97 Workshop on Intelligent Scalable Text Summarization, 82–88 Retrieved 19 Dec 2005 from the World Wide Web : <<http://www.isi.edu/~marcu/papers/summary97.ps>>

- [15] StartVBdotnet.com Website, **System Development LifeCycle**.
Browsed on the World Wide Web on 21 March 2006.
<<http://www.startvbdotnet.com/sdlc/sdlc.aspx>>
- [16] Buzzle.com Intelligent Life on the Web, **The Waterfall Model Explained**.
Browsed on the World Wide Web on 21 March 2006.
<<http://www.buzzle.com/editorials/1-5-2005-63768.asp>>
- [17] CodeBetter.com Website, **Software Development Life Cycle Models**.
Browsed on the World Wide Web on 21 March 2006.
<<http://codebetter.com/blogs/raymond.lewallen/archive/2005/07/13/129114.aspx>>