BAHASA MELAYU STEMMER

By
Subbu Valliappan
WEK010263

Under Supervision of
Ms Mangalam Sankupellay

Under Moderation of
Pn Norisma Idris

This thesis is submitted to Faculty of Computer Science and Information Technology, University of Malaya in partial fulfillment of the requirement for Bachelor of Computer Science

ABSTRACT

This report introduces the project, titled Bahasa Melayu Stemmer, and provides a detailed description of the system. There are a total of four chapters in this report, and all the chapters are dedicated entirely to the stemmer project. Each of the chapter presents an explanation of the system in a different perspective as to enhance one's understanding of this stemmer.

The main idea of this project is to build a stemmer that can stem any given words, and to develop a standard stemming algorithm in Malay language. The language of Malay is particularly chosen because there is a lack of complete stemmers that exist for this language at the moment. Another objective of this system implementation is to study more about the stemmer architecture. Unlike other stemmers, this stemmer would not use complex and difficult artificial intelligence techniques such as complicated knowledge representation and major parsing.

This stemmer has a wide variety of applications and it caters for a vast age group. It is expected that when this system is developed and fully utilized, it will benefit its users and customers tremendously with all of its powerful features. I truly hope that this report, and the stemmer system as well, will open up the eyes of researchers and bring a new dimension to the world of stemming.

ACKNOWLEDGEMENT

Here, I would like to thank everyone who gave their helping hand to me in making this report a success. Throughout the system proposal as well as for this project report, I have been fortunate to have many sources of inspiration and support. There are a number of people who gave their invaluable advice and encouragement during this report writing, and I would like to take this opportunity to thank these people.

First and foremost, I would like to convey my heartful thanks to my supervisor, Ms. Mangalam Sankupellay, for without her insightful ideas and guidance, the success of this report would not have been possible. She has truly been a motivating factor for me throughout the project. Next, I would like to extend my utmost thanks to my moderator, Pn. Norisma Idris, for sparing her precious time to be my moderator. I would like to thank her for giving valuable advice and suggestions during my viva.

A very special thanks goes to my parents and sister for their undying support shown to me throughout the project. They were always there to motivate me and pull me through the toughest hurdles in life. Thank you so much. I also would like to take this opportunity to thank everyone else who has in one way or another contributed to the success of this report.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

INTRODUCTION

This chapter features a full description of the proposed project including the definition, goals, objectives and the scope of the project. It also features the outline of the project's development schedule which has been represented as a diagram.

1.1 Project Overview

Welcome to the world of stemmer! Stemmer, as suggested by the word itself, simply means an act of stemming. Sounds simple, right? That's the title of my project, Bahasa Melayu Stemmer. The title may sound simple but actually it isn't. Anything related to language manipulation and processing is not easy at all and the same goes to my Bahasa Melayu Stemmer.

This stemmer would be able to stem Malay words, since it is a Bahasa Melayu Stemmer. What a fact! Forget about the fact, let's move on to the project description.

This proposed stemmer would be able to stem approximately around 12,000 Malay words. This includes all types of words including '*kata ganda*' which is regarded as an advantage of this system compared to other stemmers. Stemming usually is referred to as a process of affix (prefix and suffix) stripping. However, this stemmer would go beyond the limit to stem even the infixes, apart from prefixes and suffixes. Here goes another advantage of my Bahasa Melayu Stemmer.

That explains the project definition part but there's still a question left unanswered? Why do I choose the language of Malay to implement my stemmer? Why can't I develop a stemmer for English language? The answer is simple. There are a lot of stemmers out there devoted to English language but it is hard to find one that uses Malay language. English language stemmers are quite complete and thorough so there's no point of regenerating the generated stemmers. Rather it is worthy to give it a try to develop a stemmer in other languages and that is basically why I choose Malay language.

During the information gathering phase, I spent a quite number of days searching for stemmers in Malay language but I couldn't find even one, unfortunately. Most of the stemmers use English as the language but there is also some stemmers dedicated to other languages namely Japanese, Finnish, Swedish and Spanish. Malay stemmers do exist but most of them are being developed just for internal uses. Therefore, there is no coherence between these stemmers. Standard algorithm are well absent while generalized rules and techniques does not exist at all.

Looking at these problems, something struck my mind. I decided to develop a complete and standard stemmer for Malay language. By doing so, I hope that I could contribute to the IT world in Malaysia by doing something new rather than mundane. Apparently, that was the driving factor which made me to opt for this topic, Bahasa Melayu Stemmer.

1.2 Project Objective

This project would be an attempt to realize the fancies that spring up after reading the project title. I hope to come up with a complete stemmer, among the first of its kind to be developed in Malay language. This fact itself will give a new dimension to the project. In the IT scope, hopefully this stemmer would appear as one of the competitors among other available stemmers. Other objectives are as follows:

- To develop a complete stemmer which could stem any given Malay words
- To come up with a standard algorithm for developing a stemming program in Bahasa Melayu
- To improve the efficiency of stemmers that are readily available in the market nowadays, especially stemmers that uses Malay language
- To provide a general stemming engine which could be used by other applications and systems
- To be used as a global dictionary for Bahasa Melayu. Users can check whether a word is spelled correctly or not and also whether a particular word is a Malay word or not

1.3 Project Scope

Project scope is as important as project objective. Without a properly defined scope, the project to be developed is bound for failure. This is because the project scope sets the fundamental boundaries of the system. As for my project, I've chosen a scope limit which is average and acceptable, that is not too small or too big. A small scope will limit the system's functional and non-functional requirements while a big scope will be fastidious to determine the words and time consuming to be developed. The scope of my project is presented below:

- This stemmer would be able to stem Malay words only (approximately around 12,000 words)
- This stemmer would be able to stem pure Malay words only, and not borrowed Malay words
- This stemmer would be able to stem any types of Malay words including '*kata ganda*'
- This stemmer would deal with stripping of prefixes, suffixes and also infixes
- This system would be a stand-alone system and not a web-based one

1.4 Target User

This stemmer is expected to be among the first Malay language stemmers. Once the stemmer has been developed, its target user would be:

- All genre of society who would like to give the stemmer a try
- Students and researchers who would like to discover and deepen their knowledge about Malay language stemming algorithm and techniques
- Developers of other systems who are interested in using or integrating their system with my stemmer

1.5 Project Schedule

The project development schedule for this project is as follows:

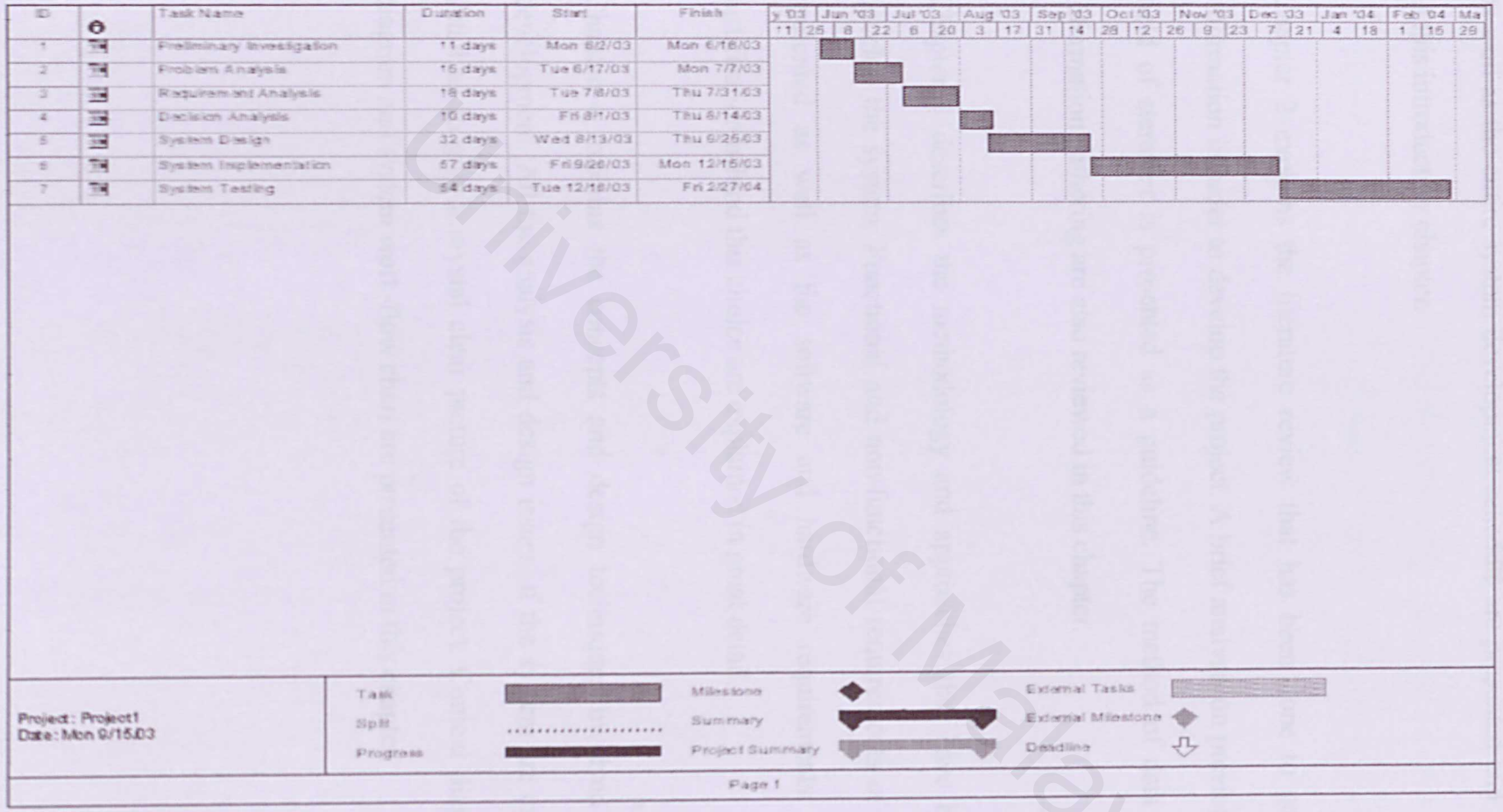


Figure 1.1 Project Development Schedule

1.6 Report Layout

Chapter 1 serves the purpose of introducing the project. Project objectives and scopes as well as the entire system development schedule are presented and fully described in this introductory chapter.

Chapter 2 explains the literature review that has been done to get the required information in order to develop the project. A brief analysis on previous works in the field of stemmer is presented as a guideline. The method of data collection and information gathering are also reviewed in this chapter.

Chapter 3 describes the methodology and approaches that have been chosen to develop the system. Functional and non-functional requirements of the system are presented as well as the software and hardware requirements. Rationale and justification behind this choice are explained in great detail.

Chapter 4 explains the concepts and design techniques involved in the system development. All the analysis and design issues of the system are explored in great detail as to give a crystal clear picture of the project. Context diagram, data-flow diagram and system work-flow chart are presented in this chapter.

1.7 Summary

From this first chapter, an overview of the proposed system, Bahasa Melayu Stemmer, is presented and fully described. The objectives, scopes and purpose of the project are explained as well. This chapter also provides information about the project development schedule. All this information is meant to serve as an introduction of the proposed system.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter will describe in detail the various studies and researches done in the topics related to stemmer. Existing stemmers would be examined and explanation would be provided based on stemming programs, algorithms, techniques and technologies. It is the objective of this chapter to outline systematically all the studies that had been carried out in this field beforehand so that it would assist in the proper development of the system. By carrying out this literature review, one could understand the strength, weakness, problems, opportunities and current issues about stemmers.

Literature review is an overall research that had been carried out on previous systems. As for my project, the literature review would be based on previous stemmers. The purposes of this literature review are:

- To carry out research and evaluate all those systems that has been developed previously. This could be helpful in identifying the systems' weaknesses so that this drawback could be rectified in the system to be developed
- To review and compare previous systems that has been developed in the related field as to analyze its pros and cons. This could serve as a guideline to the system development process

- To get a clear and better understanding about the concept and techniques used in previous systems so that the best solution could be implemented in this new system
- To collect information about the system in development as to smoothen the system planning and project progress

2.2 Natural Language Processing

As mentioned earlier, my project of Bahasa Melayu Stemmer would exhibit the function of stemming Malay words. Stemming is a part of *morphology*, a popular topic in *Natural Language Processing*. Before I go on to explain about stemming, I think it would be appropriate to get started with a brief introduction to *Natural Language Processing* as this could give an insight of what we are going to look at later on.

2.2.1 Natural Language

Computers use machine language to interact among themselves and we, as humans, use natural language to communicate among ourselves. Natural language can be defined as one of the languages which are naturally used to speak, write and communicate in our daily life. It can be any kind of languages; Malay, English, Chinese, Japanese, Hindi or Spanish. Natural language is a way of communication between humans. They could use this language to express themselves and even to solicit ideas and opinions. Probably no two people use exactly the same set of words

to communicate but then it is the language that brings them together. Here lies the great strength of the natural language – bringing humans together in this huge, huge world.

2.2.2 Natural Language Processing

We have gone through the definition part of natural language. Thus, natural language processing simply means processing or manipulating the natural language. In a more formal way, natural language processing is defined as a computational study of linguistics. In other words, it is a study of how to represent the natural language in computers which understands only machine language. Natural language processing is comprised of five important components, which are *syntax*, *semantic*, *pragmatic*, *phonology* and *morphology*. All these five components are explained briefly below.

Syntax

Syntax is referred to as rules governing formation of sentences. Generally, words are systematically ordered to form sentences and this order of sentence forming is called syntax. The syntactic structure of a Bahasa Melayu sentence is something like this:

Suzana[Kata nama khas] *membeli*[Kata kerja] *sebuah*[Kata bilangan] *kereta*[Kata nama] *baru*[Kata adjektif]

As we can see from the above sentence, each word falls in a particular word category or better known as 'parts of speech'. This word category is presented in [...] brackets.

The order of this word category would determine whether the sentence is grammatically correct or not.

As an instance, the word '*kereta*' from the above sentence could be replaced by the word '*rumah*' since both these words fall into the same word category, which is '*kata nama*'.

Following the same rule once again, the word '*membeli*' could be replaced by the word '*membaca*' and ironically, this is still a valid sentence according to the syntax rules.

Thus, syntax just deals with the rules of sentence formation grammatically, without prior consideration of whether the sentence is meaningful or not.

Semantic

Contrary to syntax, semantic deals with the meaning part of a sentence. Basically, a given sentence would carry some specific meaning and this meaning is essentially referred to as semantic. To further understand the meaning of semantic, let's look at this example:

Suzana membeli sebuah kereta baru

Suzana membaca sebuah kereta baru

As explained earlier, these two sentences are syntactically correct but semantically, they are not. Semantically, the first sentence is a valid sentence but the second one is invalid since it doesn't carry any meaning at all. Now, let's look at these two sentences:

Suzana membeli sebuah kereta baru

Suzana memandu sebuah kereta baru

Both of the sentences are semantically correct now. Finally, here is another part of semantic which should be considered:

Suzana membeli sebuah kereta baru

Sebuah kereta baru dibeli oleh Suzana

These two sentences are syntactically different but semantically they are the same since both of the sentences carry the similar meaning, literally.

To make a long story short, semantic just deals with the meaning of a sentence and not formation or structure of that sentence.

Pragmatic

Pragmatic also deals with the meaning of a sentence but in a different way compared to semantic. Pragmatic is formally defined as the study of the purposes and intentions of the language user. It focuses on speaker utterances and intentions rather than the

meaning of that utterance itself. This may sound complicated so we'll take a look at few examples to understand the concept of pragmatic in a simplified way.

Can you pass up the tutorial answers by today?

Bolehkah anda menghantar jawapan tutorial pada hari ini?

This sentence can be interpreted in two ways. One, the speaker (possibly a lecturer) is *requesting* his listeners (possibly students) to pass up their assignment by today. The other one, the speaker is *instructing* his listeners to pass up their assignments by today. Whether the speaker is *requesting* or *instructing*, this depends on his intention which could be identified by his voice flow. That is one aspect of pragmatic and now let's consider another aspect:

Can you swim?

Bolehkah anda berenang?

This sentence is clearly a question, usually asking for a clear-cut answer so that either *yes* or *no* would be valid responses. But how about this question:

Can I borrow your pen?

Bolehkah saya pinjam pen anda?

This question appears similar to the earlier posed question in terms of its syntactic structure and of the meanings of its individual words, but it is normally not a question but rather a request for action. In this situation, the speaker expects a pen to be passed and not a *yes* or *no* answer.

By looking at presented examples above, it is clear that pragmatic is not associated with the meaning of a sentence straightaway. Rather, it deals with purposes and intentions of the natural language speaker.

Phonology

So far, we have gone through the structure and the meaning of a sentence. Now, we will look into the sound and pronunciation part of a sentence, with respect to the words. Words consist of a sequence of distinctive sounds or phonemes. The rules to transform these sound into pronunciation of words is referred to as phonology. Basically, it is the study of linguistic sound which represents a particular phoneme in a given word.

For an instance, let's consider the word of *lahir*. This word consists of five 'unit' sounds which are '*l*', '*a*', '*h*', '*i*' and '*r*'. These 'unit' sounds are what we call phonemes. All the five different phonemes will correspond to a different sound and the combination of these sounds will provide the pronunciation for the word *lahir*.

It is just a brief description of phonology. Since this is very much different from what we are looking into, I won't go too detail on this topic. Let's put a full stop on phonology and move on to the next and final topic.

Morphology

Recall back that in my earlier definition, syntax is referred to as a set of rules governing formation of sentences. As for morphology, it is a set of rules that deals with the word formation. Words can have an internal structure of its own which corresponds to the ordering of letters to form words. This order of word formation is called morphology. Generally, there are two types of morphology, which are *inflectional* and *derivational* morphology. Consider the word *makan* which is used in the sentence below:

Sumita suka makan nasi goreng

Adding the *inflectional* morphology of '*me*' and '*di*' to the word of makan produces new words but with similar semantic or meaning. Example of sentences follows:

Sumita memakan nasi goreng tersebut

Nasi goreng tersebut dimakan oleh Sumita

In all of these sentences, the basic meaning of *makan*, which is 'eat', is preserved. On the other hand, adding the *derivational* morphology of '*an*' to the same word

produces a new word once again but now with a different meaning altogether. Let's have a look at this following sentence:

Sumita gemar menikmati makanan yang lazat

In the sentence above, the meaning of *makanan*, which is 'food', entirely differs from that of *makan*. That explains the derivational part of morphology.

Inflectional and derivational morphology can be prefixes, infixes or suffixes. It can be added to the start of a word, inserted into a word or added to the end of a word. Whatever it is, the meaning of a particular word is what matters the most. Therefore, morphology performs the function of governing word formation.

2.3 Stemming

Stemming – this word has been used quite a number of times so far and I would continue to do so throughout this project. The question arises now. What is stemming? Different people interpret it in a different way. Therefore, I present here definitions of stemming which has been made popular by experts in the filed of natural language processing. According to them, stemming can be defined as:

- An algorithm which reduces all words with the same root to a single form by stripping each word of its derivational and inflectional affixes [*Lovins, 1969*]

- A process of identifying morphological variants of words which have similar semantic interpretations and reducing it to its stem or root form [*Hayle, 1978*]
- A process of conflating different variants of a term to a single representative form [*Robinson, 1983*]
- A process of linguistic normalization, in which the variant forms of a word are reduced to a common form [*Arnold, 1991*]
- A computational procedure that is designed to bring together words that are semantically related, and to reduce them to a single form for retrieval purposes [*Spencer, 1995*]
- Truncation of words, disregarding of prefixes and/or suffixes [*McCord, 1999*]

To make a long story short, stemming could be defined as a process of affixes (prefixes and suffixes) removal in order to generate its root word. To better understand the concept of stemming, let's look at this example:

```
makanan
memakan
dimakan      --->    makan
termakan
pemakanan
```

Words on the left are words with affixes, either prefix or suffix, or both prefix and suffix. Taking these affixes off is called 'stemming' and the residual part is referred to as 'stem' or root word. E.g. all the above words will yield the root word of '*makan*'.

Stemming can be categorized into 4 approaches: affix removal, successor variety, table lookup and n-gram. Affix removal algorithms remove suffixes and/or prefixes from terms, leaving a stem. These algorithms sometimes also transform the resultant stem. There are 2 further subgroups under this category: longest match and simple removal. Longest match stemmers remove the longest possible string of characters from a term according to a set of rules. They can either be iterative in nature, i.e. suffixes are removed one after another, or longest match, i.e. if more than one suffix matches the end of a word, the longest one is selected. Simple removal stemmers remove only plurals from a term. It is commonly accepted that suffix stripping is a good idea.

Successor variety stemmers use the frequencies of letter sequences in a body of text as the basis of stemming. The n-gram method conflates terms based on the number of di-grams or n-grams they share. Since these two types of stemmers are quite difficult to implement, further explanation would be skipped as for the reason of simplicity. Yet there is another approach to stemming: table lookup. Terms and their corresponding stems can be stored in a table. Stemming is then done via lookups in the table.

The following figure shows the taxonomy of stemming algorithms as discussed above:

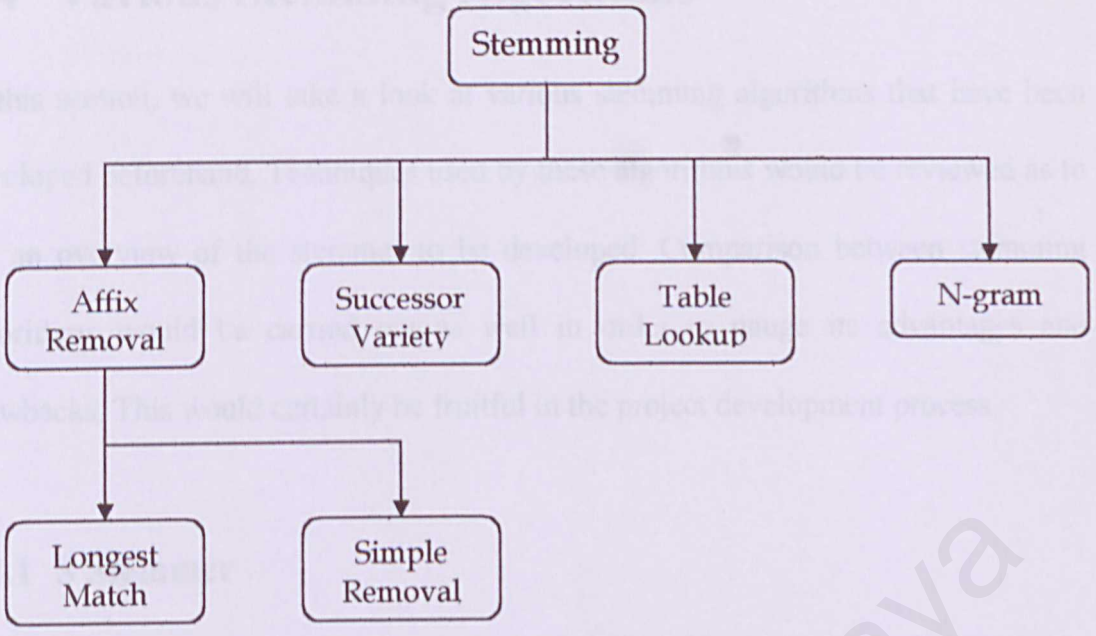


Figure 2.1 Stemming Algorithm Approaches

Stemming is used in numerous fields, in wide range of applications. However, the utmost usage of stemmer is reserved for information retrieval purposes. One of the main problems involved in the information retrieval field is the variation in word forms that is likely to be encountered (*Lennon, 1981*). The most common type of variations are spelling errors, alternative spellings, multi-word concepts, transliteration, affixes and abbreviations. One way to alleviate this problem is to use stemming. Stemming is used in these information retrieval systems to improve the matching algorithms.

In addition to that, stemmers are also being widely used in other applications as well such as automated text processing, speech synthesis and recognition, machine translation, handwriting recognition, grammar checking and sentence generation.

2.4 Various Stemming Algorithms

In this section, we will take a look at various stemming algorithms that have been developed beforehand. Techniques used by these algorithms would be reviewed as to get an overview of the stemmer to be developed. Comparison between stemming algorithms would be carried out as well in order to gauge its advantages and drawbacks. This would certainly be fruitful in the project development process.

2.4.1 S Stemmer

This is a basic stemming algorithm used in conflating singular and plural word forms, and is commonly used for minimal stemming. The rules for a version of this stemmer, shown in the table below, are only applied to words of sufficient length (3 or more characters), and are applied in an order dependent manner (that is, the first applicable rule encountered is the only used). Each rule has 3 parts; a specification of the qualifying word ending, such as "ies"; a list of exceptions; and the necessary actions.

Order	Condition	Action
1	Word ends in "ies", but not "eies" or "aies"	"ies" → "y"
2	Word ends in "es", but not "aes", "ees" or "oes"	"es" → null
3	Word ends in "s", but not "us" or "ss"	"s" → null

Table 2.1 S Stemmer Rules

Examples of S Stemmer are such as:

cities → *city*

puppies → *puppy*

buses → *bus*

branches → *branch*

books → *book*

valleys → *valley*

As suggested by its name, S Stemmer is used to stem or remove suffixes such as "ies", "es" and "s". Therefore, this stemmer is only capable in dealing with limited suffixes which is a major drawback of the stemmer.

2.4.2 Porter Stemmer

This stemming algorithm was developed by M. F. Porter in the year of 1980. It is a stemming algorithm "that looks for about 60 suffixes and involves a multi-step process that successively removes short suffixes, rather than a single removal of the longest possible suffix". The Porter algorithm consists of a set of condition/action rules. The conditions fall into 3 classes: conditions on the stem, conditions on the suffix, and conditions on the rules.

There are several types of stem conditions.

1. The *measure*, denoted *m*, of a stem is based on its alternate vowel-consonant sequences. Vowels are a, e, i, o, u, and y if preceded by a consonant. Consonants are all letters that are not vowels. Let C stand for a sequence of consonants, and V for a sequence of vowels. The measure *m*, then, is defined as $[C](VC)^{m}[V]$. The *m* in the equation indicates the number of *VC* sequences. Square brackets indicate an optional occurrence. Some examples of measures for terms follow:

Measure	Examples
0	TR, EE, TREE, Y, BY
m=1	TROUBLE, OATS, TREES, IVY
m=2	TROUBLES, PRIVATE, OATEN

Table 2.2 Measures used in Porter Stemmer

2. *<X> - the stem ends with a given letter X
3. *v - the stem contains a vowel
4. *d - the stem ends in a double consonant
5. *o - the stem ends with a consonant-vowel-consonant sequence, where the final consonant is not w, x, or y.

For example, the condition part (*d and not (*<L> or *<S> or *<Z>)) tests for a stem ending with a double consonant other than L, S, or Z.

The rules are divided into steps. The rules in a step are examined in sequence, and only one rule from a step can apply. The longest possible suffix is always removed because of the ordering of the rules within a step. The algorithm is as follows:

```
{
        step1a (word);
        step1b (stem);
        if (the second or third rule of step1b was used)
                step1b1 (stem);
        step1c (stem);
        step2 (stem);
        step3 (stem);
        step4 (stem);
        step5a (stem);
        step5b (stem);
}
```

The rules for the steps of the stemmer are as follows:

Step 1a Rules

Conditions	Suffix	Replacement	Examples
NULL	sses	ss	caresses → caress
NULL	ies	i	ponies → poni
NULL	ss	ss	ties → tie caress → caress
NULL	s	NULL	cats → cat

Table 2.3 Porter Stemmer Algorithm – Rule 1a

Step 1b Rules

Conditions	Suffix	Replacement	Examples
(m>0)	eed	ee	feed → feed agreed → agree
(*v*)	ed	NULL	plastered → plaster bled → bled
(*v*)	ing	NULL	motoring → motor sing → sing

Table 2.4 Porter Stemmer Algorithm – Rule 1b

Step 1b1 Rules

Conditions	Suffix	Replacement	Examples
NULL	at	ate	conflat(ed) → conflate
NULL	bl	ble	troubl(ing) → trouble
NULL	iz	ize	siz(ed) → size
(*d and not (*<L> or *<S> or *<Z>))	NULL	single letter	hopp(ing) → hop tann(ed) → tan fall(ing) → fall hiss(ing) → hiss fizz(ed) → fizz
(m=1 and *o)	NULL	e	fail(ing) → fail fil(ing) → file

Table 2.5 Porter Stemmer Algorithm – Rule 1b1

The rule to map to a single letter causes the removal of one of the double letter pair. The -e is put back on -at, -bl, and -iz, so that the suffixes -ate, -ble, and -ize can be recognized later. This "e" may be removed in Step 4.

Step 1c Rules

Conditions	Suffix	Replacement	Examples
(*v*)	y	i	happy → happi sky → sky

Table 2.6 Porter Stemmer Algorithm – Rule 1c

Step 1 deals with plurals and past participles. The subsequent steps are much more straightforward.

Step 2 Rules

Conditions	Suffix	Replacement	Examples
(m>0)	ational	ate	relational → relate
(m>0)	tional	tion	conditional → condition rational → rational
(m>0)	enci	ence	valenci → valence
(m>0)	anci	ance	hesitanci → hesitance
(m>0)	izer	ize	digitizer → digitize
(m>0)	abli	able	conformabli → conformable
(m>0)	alli	al	radicalli → radical
(m>0)	entli	ent	differentli →different
(m>0)	eli	e	vileli → vile
(m>0)	ousli	ous	analogousli → analogous
(m>0)	ization	ize	vietnamization → vietnamize
(m>0)	ation	ate	predication → predicate
(m>0)	ator	ate	operator → operate
(m>0)	alism	al	feudalism → feudal
(m>0)	iveness	ive	decisiveness → decisive
(m>0)	fullness	ful	hopefulness → hopeful
(m>0)	ousness	ous	callousness → callous
(m>0)	aliti	al	formality → formal
(m>0)	iviti	ive	sensitiviti → sensitive
(m>0)	biliti	ble	sensibiliti → sensible

Table 2.7 Porter Stemmer Algorithm – Rule 2

Step 3 Rules

Conditions	Suffix	Replacement	Examples
(m>0)	icate	ic	triplicate → triplic
(m>0)	ative	NULL	formative → form
(m>0)	alize	al	formalize → formal
(m>0)	iciti	ic	electriciti → electric
(m>0)	ical	ic	electrical → electric
(m>0)	ful	NULL	hopeful → hope
(m>0)	ness	NULL	goodness → good

Table 2.8 Porter Stemmer Algorithm – Rule 3

Step 4 Rules

Conditions	Suffix	Replacement	Examples
(m>1)	al	NULL	revival → reviv
(m>1)	ance	NULL	allowance → allow
(m>1)	ence	NULL	inference → infer
(m>1)	er	NULL	airliner → airlin
(m>1)	ic	NULL	gyroscopic → gyroscop
(m>1)	able	NULL	adjustable → adjust
(m>1)	ible	NULL	defensible → defens
(m>1)	ant	NULL	irritant → irrit
(m>1)	ement	NULL	replacement → replac
(m>1)	ment	NULL	adjustment → adjust
(m>1)	ent	NULL	dependent → depend
(m>1 and (*<S> or *<T>))	ion	NULL	adoption → adopt
(m>1)	ou	NULL	homologou → homolog
(m>1)	ism	NULL	communism → commun
(m>1)	ate	NULL	activate → activ
(m>1)	iti	NULL	angulariti → angular
(m>1)	ous	NULL	homologous → homolog
(m>1)	ive	NULL	effective → effect
(m>1)	ize	NULL	bowdlerize → bowdler

Table 2.9 Porter Stemmer Algorithm – Rule 4

The suffixes are now removed. All that remains is a little tidying up.

Step 5a Rules

Conditions	Suffix	Replacement	Examples
(m>1)	e	NULL	probate → probat rate → rate
(m=1 and not *o)	e	NULL	cease → ceas

Table 2.10 Porter Stemmer Algorithm – Rule 5a

Step 5b Rules

Conditions	Suffix	Replacement	Examples
(m>1 and *d and *<L>)	NULL	single letter	control → control roll → roll

Table 2.11 Porter Stemmer Algorithm – Rule 5b

The algorithm is careful not to remove a suffix when the stem is too short, the length of the stem being given by its measure, m. There is no linguistic basis for this approach. It was merely observed that m could be used quite effectively to help decide whether or not it was wise to take off a suffix. Here lies the power of this algorithm but then it comes along with a cost. No matter how powerful it is, this algorithm just could be used to stem or remove suffixes and doesn't deal with prefixes and infixes.

2.4.3 Paice-Husk Stemmer

The Paice/Husk Stemmer is iterative, and uses just one table of rules; each rule may specify either deletion or replacement of an ending. The rules are grouped into sections corresponding to the final letter of the suffix; this means that the rule table is accessed quickly by looking up the final letter of the current word or truncated word. Within each section, the ordering of the rules is significant. Some rules are restricted

to 'intact' words – i.e. words from which no ending has yet been removed. A simple blanket acceptability test is applied before any matching rule is activated. After a rule has been applied, processing may be allowed to continue iteratively, or may be terminated.

The Rule Format

In the following, the term *form* refers to any word or part-word which is being considered for stemming. The original word before any changes have been made to it is said to be *intact*. Each line in the rule table holds a separate stemming rule. Braces {...} enclose comments describing the action of each rule.

The rules in the table are grouped into *sections*, each containing all those rules relating to a particular final letter (known as the *section letter*).

Each rule has five components, two of which are optional:

- an *ending* of one or more characters, held in reverse order;
- an optional *intact flag* "*";
- a digit specifying the *remove total* (may be zero);
- an optional *append string* of one or more characters;
- a *continuation symbol*, ">" or ".".

Within each section, the order of the rules is significant.

Example 1:

The rule "sei3y>" means: if the word ends in "-ies" then replace the last three letters by "-y" and then apply the stemmer again to the truncated form.

Example 2:

The rule "mu*2." means: if the word ends in "-um" and if the word is intact, then remove the last two letters and terminate. This converts "maximum" to "maxim" but leaves "presum" (from "presumably") unchanged.

Example 3:

The rule "ylp0." means: if the word ends in "-ply" then leave it unchanged and terminate. This ensures that the subsequent rule "yl2>" does not remove the "-ly" from "multiply".

Example 4:

The rule "nois4j>" causes "-sion" endings to be replaced by "-j". This acts as a dummy, causing activation of the "j" section of the rules (q.v.). Hence "provision" is converted first to "provij" and then to "provid".

The Algorithm

1. Select relevant section:

   Inspect the final letter of the form;

   If there is no section corresponding to that letter,

      Then terminate;

   Otherwise,

      Consider the first rule in the relevant section.

2. Check applicability of rule:

   If the final letters of the form do not match the reversed ending in the rule,

      Then go to 4;

   If the ending matches, and the intact flag is set, and the form is not intact,

      Then go to 4;

   If the acceptability conditions (see below) are not satisfied,

      Then go to 4.

3. Apply rule:

   Delete from the right end of the form the number of characters specified by the remove total;

   If there is an append string,

      Then append it to the form;

   If the continuation symbol is ".",

      Then terminate;

If the continuation symbol is ">",

Then go to 1.

4. Look for another rule:

Move to the next rule in the table;

If the section letter has changed,

Then terminate;

Otherwise,

Go to 2.

Acceptability Conditions

If these conditions were not present, the words "rent", "rant", "rice", "rage", "rise", "rate", "ration" and "river" would be reduced to "r" by the rules shown. The conditions used to overcome this problem are:

a) if the form starts with a vowel then at least two letter must remain after stemming (e.g., "owed" / "owing" → "ow", but not "ear" → "e").

b) if the form starts with a consonant then at least three letters must remain after stemming and at least one of these must be a vowel or "y" (e.g., "saying" → "say" and "crying" → "cry", but not "string" → "str", "meant" → "me" or "cement" → "ce").

These conditions wrongly prevent the stemming of various short-rooted words (e.g., "doing", "dying", "being"); it is probably best to deal with these words separately by lexical lookup.

2.4.4 Table Lookup

Table Lookup is a straightforward stemming algorithm. It requires the storage of a table of all index terms and their stems. For example:

Term	Stem
educated	educate
education	educate
educational	educate

Table 2.12 An Example of Table Lookup

Terms from queries and indexes could then be stemmed via a simple table lookup. Using a B-tree or hash table, such lookups would be very fast. Obviously, this is a simple and faster way of implementing a stemmer. However, there are problems with this approach. The first is that there is no complete data exist for a language. Even if there were, many terms found in databases would not be represented since they are domain dependent that is not standard in a particular language. For these terms, some other stemming methods would be required. Another problem is the storage overhead for such a table, though trading space for time is sometimes warranted. Last but not least, this method is too tedious to be performed and is an inefficient way of stemming.

2.5 Previous Stemmers

Various stemming algorithms have been explored in the previous section and in this section, the focus is on the stemmers. Previously developed stemmers as well as currently existing stemmers would be analyzed and reviewed as to gauge the required information. Knowledge about previous related systems is necessary for a successful development process and this is what being attempted in this section.

2.5.1 Nice Stemmer

Nice Stemmer is a stemmer that has been developed to handle English words. The developers of this stemmer are Kiduk Yang, Denqi Song, Rong Tang and Wooseob Jeoung. It is regarded as one of the complete ever stemmer that have been developed in the stemming field. Not only complete, it is complex enough to deal with. Nice stemmer is composed of four distinctive stemmers and each of this stemmer handles a certain part of the stemming process. These four stemmers are described below:

- Simple Stemmer - simple plural removal
- Porter Stemmer - morphological variations
- Inflectional Stemmer - irregulated plural form
- Combination Stemmer - stem the word simultaneously

These four stemmers combine together to form the major stemmer, called Nice Stemmer. The important features of this stemmer are:

- Nice Stemmer serves the user with a variety of needs: the system accepts three forms of input: words, files as well as index files
- Nice Stemmer provides users with four stemming systems, users may select the appropriate stemmer based on their particular needs

Each stemmer has its inadequacy in its algorithm and quality of performance, the invention of the combination stemmer is originated from the idea of aggregating the strength of Simple, Porter and Inflectional stemmers, thereby to improve the quality of performance. Nice Stemmer is nice not only because it does not impose a single stemmer system to the user, but also because it presents its own contribution to the field of stemming. The combination stemmer is far from perfect, but since it is intended to adapt the good aspects of three existing stemmers, it becomes a product that signifies the advancement of stemming in the field of stemming.

Nice Stemmer is accessible at the following URL:

http://ils.unc.edu/iris/nstem.htm

2.5.2 Text Stemmer

Another type of stemmer that would be reviewed here is Text Stemmer. This stemmer has been developed by Christopher Fox and Brian Fox. It is a trivial English stemmer that has been designed to handle character encoding of the target language as well as to deal with morphological and orthographical variations. Its main objective is to provide an easy-to-use stemming system that mimics the way a person narrow down and refines his search for a particular root word.

Text Stemmer has created a program to generate stemmers from stemmer specification files. This approach to stemmer implementation has two advantages:

- Simple textual stemmer specification files are used to specify the stemmers to be generated. It is much easier to create and modify stemmers this way than it is to write custom stemmer code (the traditional way to create a stemmer). Thus stemmer generation is human resource efficient
- Generated stemmers use finite state machines to do stemming, which is very fast. Generated stemmers are usually faster than all but the most highly optimized custom stemmer programs

This stemmer would be able to stem any type of text and this is probably why it is named as Text Stemmer. Types of text ranges from a single word to even dozens of files. Thus, this stemmer is capable of stemming the entire words in a large database, providing faster and better stemming performance. The stemming process itself is performed in an efficient way as to maintain an uniformed stemmer generation process.

Text Stemmer is accessible at the following URL:

http://rayuela.ieec.uned.es/cgi-bin/ircourse/stem2.perl

2.6 Fact-Finding Techniques

Generally, system development will not be complete without information gathering or data collection. Thus, the gathered information is vital for the system to achieve its vision and goals. This information can be obtained through several sources and each source give different level of information. Several methods have been used for the purpose of research in this literature review section. Among the methods are:

- Internet Surfing - Internet has been a lot of help in this stemmer development. By surfing the internet, a wide variety of information can be obtained and all these information are very useful for the research towards the existing system

- Documentation - Analysis and research has been conducted on the existing and available documents which have related topics with the system to be developed. This is done to better the results and findings obtained from the research method

- Interview - This method provides an opportunity to meet some experienced staffs in the related field and discuss certain topics and issues with them face-to-face. This technique is basically used to verify facts and solicit opinions

2.7 Summary

This chapter outlined and described in detail the various topics and issues researched throughout the project. A thorough and comprehensive analysis has been carried out on previous stemmers. As for the gathered information, it was derived mainly from books and websites, which are both popular and reliable.

The research was important in a way to enable better understanding of the different stemmers that had been developed previously. This allows the best combination of algorithms, techniques and technologies to be selected to design, develop and implement this stemmer project. However, it is to be mentioned that there remain many more stemmer systems that were not covered in this literature review chapter. The topics covered here were only some of the more popular stemmers. Nevertheless, the research done was sufficient for the purpose of choosing the appropriate tools to develop and implement this stemmer project.

The next chapter, methodology, will attempt to explain in detail the methods to be used in developing my project, Bahasa Melayu Stemmer.

CHAPTER THREE

METHODOLOGY

3.1 Introduction

Developing a computer system has never been an easy task. There are many facets involved in the system development process, starting from information gathering to system implementation and even maintenance. Usually, the risk of failure for any given computer system is relatively high. Therefore, proper planning of the project and most importantly, appropriate methodology must be adopted to produce the desired outcome. This chapter will draw out the suitable methodology for the proposed system along with the stages involved. Apart from that, system requirements and development tools would be discussed in this chapter, too.

3.2 System Methodology

Methodology refers to the science of how a system is developed. Generally, choosing the appropriate methodology will determine the success of a system. As for this stemmer project, a methodology is selected based on the congruence between the method and the system to be developed. The phases of the chosen methodology and its advantages are discussed in this section.

Methodology Chosen – Waterfall Model with Prototyping

The development methodology for this project is based on the System Development Life Cycle (SDLC). The model chosen is the 'waterfall model with prototyping'. This model is a variation, or to be exact, is an enhancement of 'waterfall model'. Before I go on to explain the chosen model (waterfall model with prototyping) in great detail, let's have a look at the waterfall model first.

Waterfall model is the first ever published model of the software development process (Royce, 1970). The waterfall model is illustrated in Figure 3.1.

As we know, system development generally passes through a series of phases or stages. The stages in the waterfall model are depicted as cascading from one to another and this is probably why it is called as the 'waterfall model'. As the figure implies, a given development stage have to be completed before the next stage could begin. Each phase in the waterfall model is presented discretely and never accomplished as a separate step. Several activities can occur simultaneously and these activities may be repeated a number of times to achieve the desired output.

Thus, the waterfall model presents a very high level of what goes on during development and it also suggests to developers the sequence of events they should expect to encounter. The principal stages of the model map onto fundamental development activities. It can also be useful in helping the developers to layout what they need to do and what are the tasks that must be focused on. Besides that, the

developers could use this model to gauge how close the project is to completion at a given point of time.

There are seven different stages in a waterfall model and these stages are:

- Requirements Analysis
- System and Software Design
- Implementation
- Unit and Integration Testing
- System Testing
- Acceptance Testing
- Operation and Maintenance

As discussed earlier, all these stages are cascaded from one to another. A given stage should not start until the previous phase has been completed. In practice, these stages overlap and feed information to each other. Thus, communication between stages does exist and this provides a better understanding of developing a system.

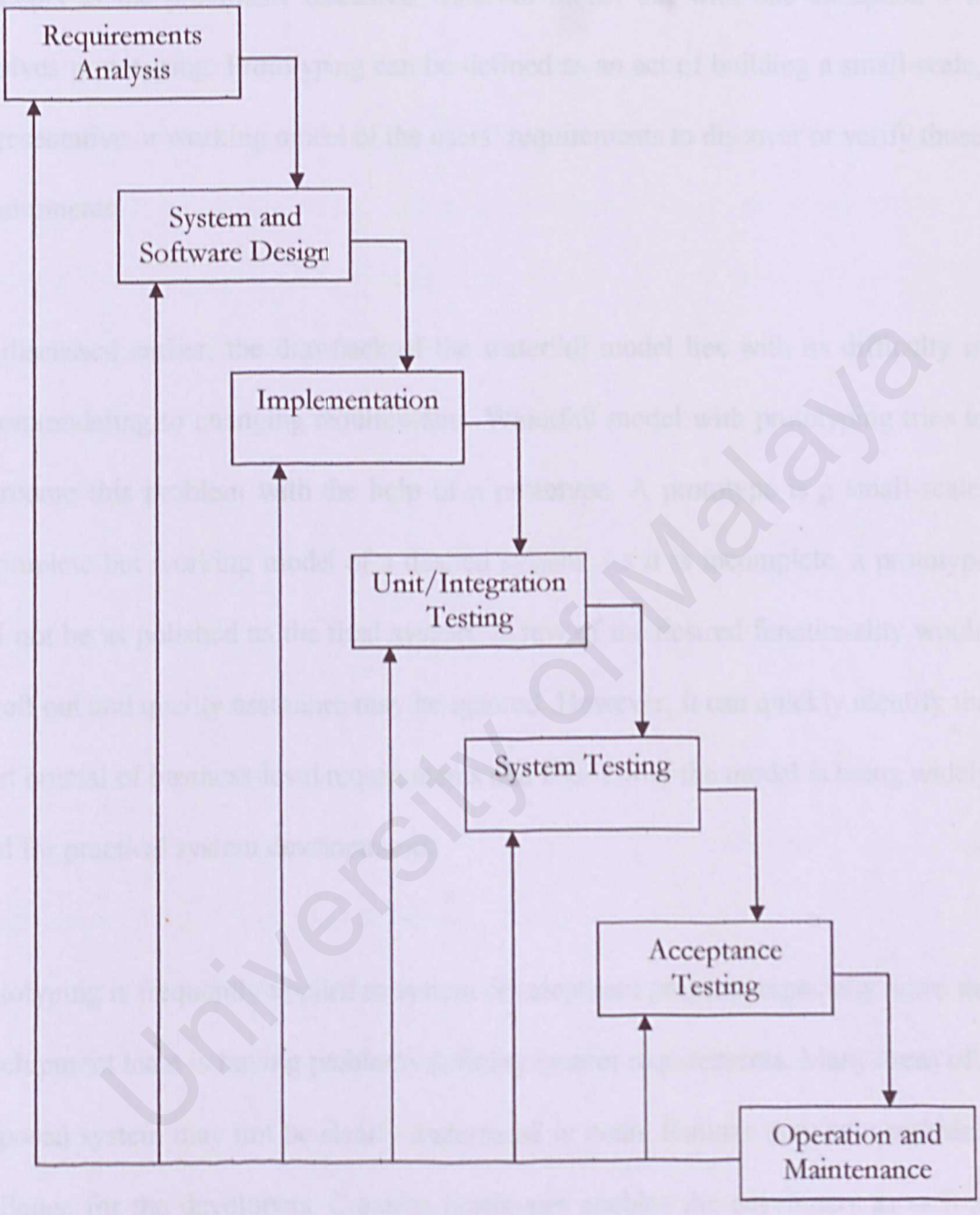


Figure 3.1 Waterfall Model

Now, let's look at the waterfall model with prototyping. As mentioned earlier, this model is an enhancement of the waterfall model. This model inherits the same concepts as the previously discussed waterfall model but with one exception – it involves prototyping. Prototyping can be defined as an act of building a small-scale, representative or working model of the users' requirements to discover or verify those requirements.

As discussed earlier, the drawback of the waterfall model lies with its difficulty in accommodating to changing requirements. Waterfall model with prototyping tries to overcome this problem with the help of a prototype. A prototype is a small-scale, incomplete but working model of a desired system. As it is incomplete, a prototype will not be as polished as the final system. A few of the desired functionality would be left out and quality assurance may be ignored. However, it can quickly identify the most crucial of business-level requirements and this is why the model is being widely used for practical system developments.

Prototyping is frequently applied to system development projects, especially when the development team is having problems defining system requirements. Many areas of a proposed system may not be clearly understood or some features may be a technical challenge for the developers. Creating prototypes enables the developers as well as the users to better understand the issues involved with developing the system. This technique minimizes the risk of a system being delivered that doesn't meet user needs or one that can't fulfill technical requirements.

Prototyping is actually an external process and it has its own development cycle, which will be carried out earlier in the system development process. Prototyping makes use of prototype; a partially developed product that enables customers and developers to examine some aspect of the proposed system. Usually, only the areas where the requirements are not clearly understood are prototyped and this prototyping process would be performed in an iterative manner. Each prototype model is reviewed by end-users and management, who make recommendations about requirements, methods and formats. The prototype is then corrected, enhanced or refined to reflect the new requirements. The refinement process continues until the prototype is accepted by both the users and developers.

There are two processes involved in a waterfall model with prototyping; validation and verification. Both these processes are performed in the system testing phase. Validation ensures that the system has implemented all of the requirements, so that each system function can be traced back to a particular requirement in the specification. Verification ensures that each system function works correctly and as needed. That is, validation makes sure that the developer is building the right product according to the specification whereas verification checks the quality of the implementation.

The waterfall model with prototyping which has been adapted in the development process of this stemmer project is as illustrated below:

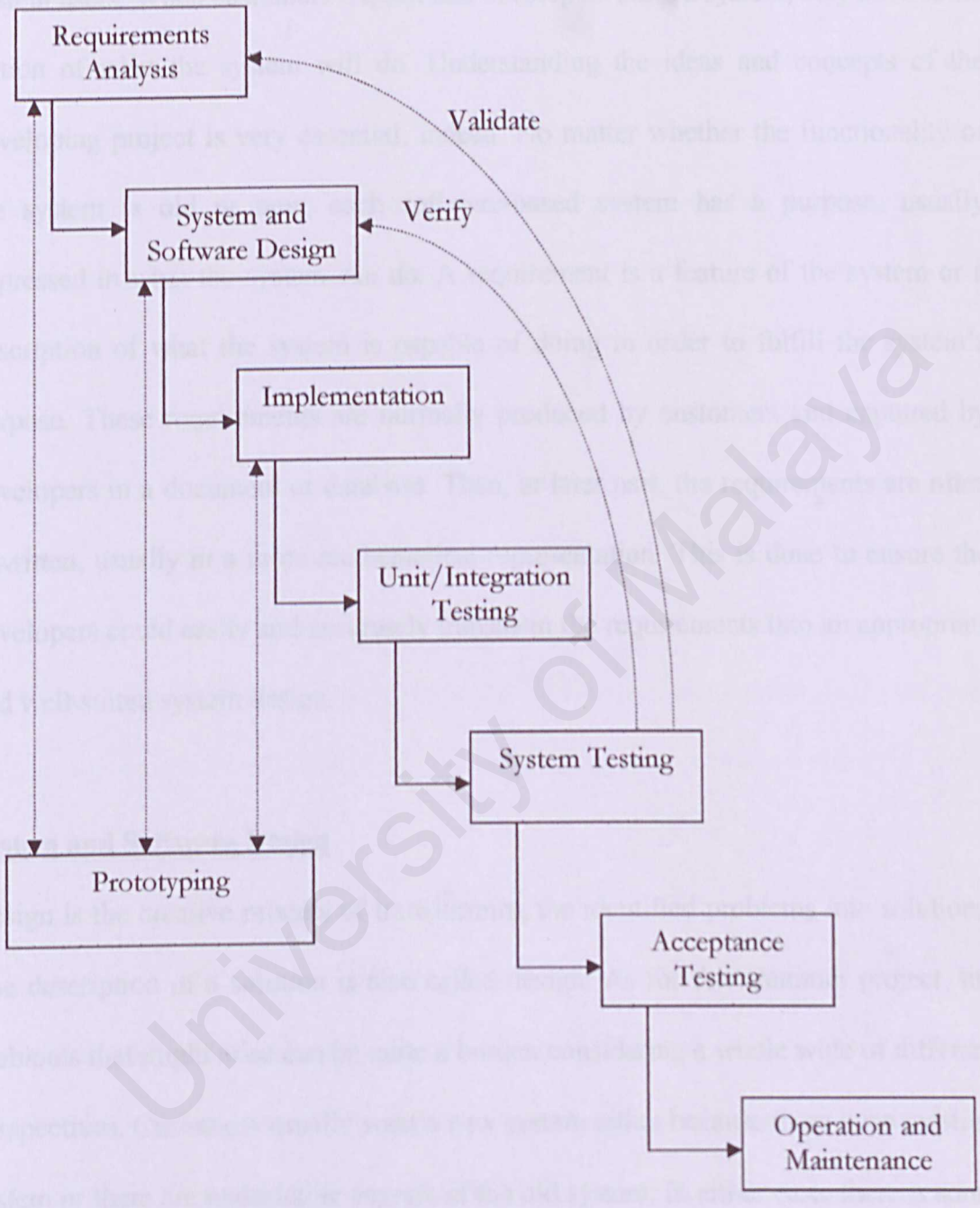


Figure 3.2 Waterfall Model with Prototyping

Requirements Analysis

The system's services, constraints and goals are established by consultation with system users. When customers request that developers build a system, they have some notion of what the system will do. Understanding the ideas and concepts of the developing project is very essential, indeed. No matter whether the functionality of the system is old or new, each software-based system has a purpose, usually expressed in what the system can do. A requirement is a feature of the system or a description of what the system is capable of doing in order to fulfill the system's purpose. These requirements are normally produced by customers and captured by developers in a document or database. Then, at later part, the requirements are often rewritten, usually in a more mathematical representation. This is done to ensure the developers could easily and accurately transform the requirements into an appropriate and well-suited system design.

System and Software Design

Design is the creative process of transforming the identified problems into solutions. The description of a solution is also called design. As for this stemmer project, the problems that might arise can be quite a burden considering a whole wide of different perspectives. Customers usually want a new system either because there is no existing system or there are undesirable aspects of the old system. In either case, there is some kind of problems that need to be solved. Thus, system design is the best way to find solutions out of the problems being faced. The system design process partitions the requirements to either hardware or software systems. Meanwhile, software design

involves identifying and describing the fundamental software system abstractions and their relationships. These two types of design combine and integrate with each other to establish an overall system architecture.

Implementation

Implementation merely refers to the process of converting a system specification into an executable system. During this stage, the software design is realized as a set of programs or program units. Thus, programming or program coding is an important task which is closely related to implementation stage. Coding involves algorithms and data structures being used in the development process. Choosing the right programming language is an important aspect of coding which determines the success of the system being developed. System components such as database, interface and other modules are programmed or constructed in this implementation stage. Thus, it is clear that this stage is very essential in the whole system development process.

Unit and Integration Testing

System development usually involves several stages of testing. Unit testing is the first stage. Unit testing involves verifying that each unit of the developed system meets its specification. Individual components are tested to ensure that they operate correctly and accurately. Each component is tested independently, without other system components. Unit testing is done in a controlled environment whenever possible, so that the test team can feed a predetermined set of data to the component being tested. The output actions and the produced data are observed and evaluated. In addition, the

test team checks the internal data structures, logic and boundary conditions for the input and output data. When collections of components have been unit-tested, the next step is to ensure that the interfaces and communications among the components are defined and handled properly. This is done through another type of testing, called integration testing. This involves testing groups or collections of system components which have been integrated into sub-systems. Integration testing usually is used to verify that the developed system components work together as described in the system specification.

System Testing

The sub-systems are integrated to make up the system and this system should be tested before delivered to the customer. Testing the system is very different from other types of testing. In unit and integration testing, just a few of the development team members would be involved and they would be using individual or separate test data and test cases. Whereas for a system testing, it involves the entire development team which should carry out the testing process in a coordinated way. System testing process is concerned with finding errors that result from unanticipated interactions between sub-systems and their interface problems. It is also concerned with validating that the developed system meets its functional and non-functional requirements.

Acceptance Testing

So far, all the tests that have been performed are carried out by the developers, based on their understanding of the system and its objectives. However, the system should

be tested by customers as well since the developed system are meant for them and should satisfy their needs and requirements. It is advisable for customers to test the system to ensure that it meets their understanding of the requirements, which may be different from the developers'. In this acceptance test, the system is tested with data supplied by the customers rather than simulated test data. This could reveal errors and omissions in the system requirements definition because the real data exercise the system in different ways from the test data. Acceptance testing may also reveal requirements problems where the system's facilities do not really meet the user's needs or the system performance is quite unacceptable. For this reason, a final installation test may be performed to allow users to exercise system functions and document additional problems.

Operation and Maintenance

System development is considered to be a creative activity where a software system is developed from an initial concept through to a working system. At each stage of system development, works produced at earlier stages are continually referred. The design components are tied up to the requirements specification while the code components are reviewed and cross-referenced for compliance with the design. Thus, system development involves looking back at the development process that has been carried out in a careful, controlled way. System maintenance is different. It involves looking forward upon the system development by establishing a working relationship with the system customers, users and operators to find out how satisfied they are with the system. Problems could be anticipated and functional changes could be

considered as a maintenance step to meet the changing business requirements. Thus, system maintenance is the process of changing a developed system once it has gone into operational use. Although the costs of maintenance are often several times the initial development costs, maintenance processes are considered to be less challenging than original software development process.

There are several advantages in using the waterfall model with prototyping:

- ✓ Reduces risk and uncertainty in the development process
- ✓ Understands the feasibility of a design or approach
- ✓ Allows all or part of the system to be constructed quickly
- ✓ User needs and requirements are better accommodated
- ✓ The resulting system is easier to use and maintain
- ✓ The final design is of highest quality

As we have discussed above, waterfall model with prototyping is one of the earliest model that have been developed in the field of software development. Apart from that, it is also the easiest development model that one could find. Its simplicity makes it easy to explain to users and customers who are not familiar with software development. Developers also would not face much problem by adapting to this model since it has a simple architecture. This waterfall model with prototyping normally would be used when the requirements are well understood and I have chosen this methodology for the same reason – my stemmer project's requirements are crystal-clear and well understood.

3.3 Stemming Algorithm

Stemmers need a variety of requirements in order to be successfully implemented. The top most requirement would be the words, since a stemmer can't operate without it. Words are needed to perform stemming but it is not all a stemmer needs. Manipulation of words is as important as the words itself and this manipulation is provided by a method or algorithm. As in this case, it is called stemming algorithm.

Stemming Algorithm Chosen – Porter Algorithm

Several stemming algorithms have been discussed in the previous chapter. Each of this algorithm has its pros and cons. The task now is to choose a suitable stemming algorithm that fits exactly with this stemmer system. After a thorough analysis, I have compared all the presented stemming algorithms and finally, opted for Porter Algorithm. As described earlier, it is a stemming algorithm that involves a multi-step process that successfully removes short suffixes, rather than a single removal of the longest possible suffix. The algorithm is careful not to remove a suffix when the stem is too short and this certainly helps to improve the performance of the resultant stemmer.

However, the actual algorithm was meant for English language but this stemmer system would be implemented in Malay language. Language differs, from English to Malay, and this results in the need for some variations in the stemming algorithm as well. There is a slight difference between word structures in Malay and English. These differences should be taken into account when designing or developing the

stemmer project. The original Porter Stemming Algorithm should be modified accordingly as to adapt to the changing nature of the language.

Hence, the actual Porter Algorithm would be used just as a guideline in this development process, and have not been adapted totally. As a replacement, a modified version of the Porter Stemming Algorithm would be used to implement this stemmer project and an overview of the algorithm is presented here:

```
{
        Step1 (word);
        Step2a (stem_suffix);
        Step2b (stem_prefix);
        Step3a (stem_suffix);
        Step3b (stem_prefix);
        Step4 (stem_infix);
        Step5 (root_word);
}
```

3.4 Sentence Parsing

Stemming can be done on individual words but it is not the case with sentences. A given sentence can't be stemmed straightaway. Nevertheless, the sentence should be broken down to words before stemming could be carried out. Sentence parsing is the technique used to split a sentence into its words. In this section, I will attempt to give a brief description of sentence parsing and its importance to my stemmer project.

By implementing parsing, a sentence can be broken down into its words and this is done with accordance to the type of words. That is, a sentence parser would split a sentence by classifying words into their types. Then, each of the classified words would be mapped to a structure called parse tree. This parse tree reveals the structure of sentence formation as well as the meaning of the sentence itself.

The type of a particular word could be identified by a sentence parser and this is useful in analyzing sentences. For an example, a sentence in Malay language would be normally comprised of a noun phrase (NP) and a verb phrase (VP), which can be represented as:

$$S \rightarrow NP + VP$$

A noun phrase further can be decomposed into either a single noun (N) or a combination of a noun (N) and a determiner (DET). Another possibility of a noun

phrase would be the sequence of a noun (N), an adjective (ADJ) and a determiner (DET). This representation can be shown as:

NP → N

NP → N + DET

NP → N + ADJ + DET

In a similar way, a verb phrase could be decomposed into three possibilities; either a combination of a verb (V) and a noun (N), a verb (V) and a noun phrase (NP) or a verb (V), a noun (N) and a prepositional phrase (PP). This is represented as:

VP → V + N

VP → V + NP

VP → V + N + PP

A prepositional phrase is a combination of a preposition (P) and a noun (N), which can be represented as:

PP → P + N

For an improved understanding on sentence parsing, let's look at the sentence parser below which is represented as a parse tree:

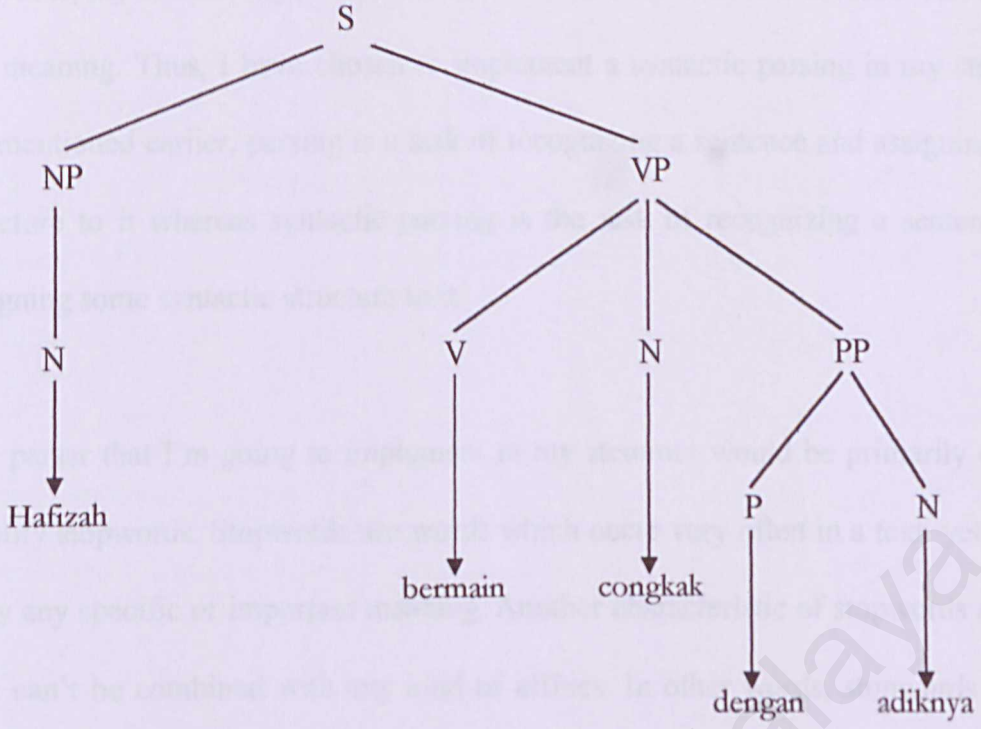


Figure 3.3 Sentence Parser (Parse Tree)

The primary job of a sentence parser is to identify the types of words in a sentence and then to classify them accordingly. The above example illustrates a parse tree for the sentence '*Hafizah bermain congkak dengan adiknya*'. Now, let's trace back the parsing process done to that sentence. The parser starts off with the first word of the sentence, that is '*Hafizah*'. Since this is a name of a person, the word is classified as a noun. Then, the parser moves on to the second word. The word of '*bermain*' is classified as a verb since it is an act performed by someone. The parsing process continues in the same way until the end of the sentence is reached.

So far, we have gone through the actual process of sentence parsing. This is often called as semantic parsing; parsing which focuses on the meaning of sentences.

However, my stemmer system just needs to handle the structure of sentences, and not the meaning. Thus, I have chosen to implement a syntactic parsing in my stemmer. As mentioned earlier, parsing is a task of recognizing a sentence and assigning some structure to it whereas syntactic parsing is the task of recognizing a sentence and assigning some syntactic structure to it.

The parser that I'm going to implement in my stemmer would be primarily used to identify stopwords. Stopwords are words which occur very often in a text, yet do not carry any specific or important meaning. Another characteristic of stopwords are that they can't be combined with any kind of affixes. In other words, stopwords appear just as root words. Therefore, the stopwords which have been identified by the parser will be presented as output straightaway, without entering the stemming process. All the other words, non-stopwords, would be passed on to the stemming engine. This type of parsing is chosen because it would most probably reduce the response time of the system and hopefully, the performance of the stemmer would improve as well.

3.5 System Requirements

The descriptions of the services and constraints which would be provided by the system are usually referred to as system requirements. System requirements can be a high-level, abstract statement or even a detailed, mathematically formal definition. No matter how it is represented, the objective of system requirements remains the same – to describe the service that the system should provide or a constraint on the system. Basically, system requirements are often classified as functional or non-functional requirements. Both of these requirements are described in this section.

3.5.1 Functional Requirements

Functional requirements for a system describe the functionality or services that the system is expected to provide. It is often a combination of all the main modules of a system, which are made to communicate internally or externally. The functional requirements for this Bahasa Melayu Stemmer are as follows:

- **Word Dictionary**

  The system should have a database of Malay words as to facilitate the process of stemming. All the root words from the Malay dictionary (approximately 12,000 words) would be stored in a text file, which acts as a database for this stemmer project. This word dictionary would then be used to determine whether a given word is in its root form or not. Stemming would be carried out in an iterative way and at the end of each iterative, the resultant word

would be compared with the words in the word dictionary to determine whether the desired output (root word) has been achieved.

❖ **Stemming Engine**

This is the real 'engine' of this stemmer project. The real processing (stemming) would be performed in this module, called the stemming engine. Therefore, the performance of this stemmer is totally dependent on its stemming engine. There are basically three types of stemming which could be performed; prefix removal, suffix removal and infix removal. This stemming engine would be able to handle all three types of stemming. As mentioned earlier, the stemming process is an iterative one and each iterative would try to remove a group of words from a given word. This iteration would continue until it reaches the point where it should stop – when the root word has been achieved.

❖ **Exception List**

There is another important aspect of stemming which should be paid attention to. Most of the things in the world are not clear-cut or straightforward, and the same goes to this stemmer project. The process of stemming does not follow a general rule all the time and for all the words. In some cases, the stemming process does involve a quite number of variations and these exceptions are the one I'm trying to overcome – by introducing an exception list to this stemmer.

3.5.2 Non-Functional Requirements

Non-functional requirements, as the name suggests, are those requirements which are not directly concerned with the specific functions delivered by a system. On the other way, non-functional requirements are the constraints under which a system must operate and the standards that must be met by the delivered system. The non-functional requirements for this Bahasa Melayu Stemmer are as follows:

- **Efficiency**

  The proposed stemmer project should be able to stem a given word for an unlimited number of times to produce similar outcomes. This is a must as to ensure the performance and efficiency of the stemmer.

- **Simplicity**

  Simplicity is the best policy. The stemming program as well as the interface should be developed in the simplest way possible without affecting its efficiency or performance as this would help to keep the user attention focused.

- **Reliability**

  This stemmer should be reliable in the sense of its performance. It should not produce dangerous or costly failures when it is highly relied upon. The recovery from a hang or crash should be reasonable as well.

- **Maintainability**

  The program codes have to be written in such a way that it could be changed upon in the future. The coding of the program should be easy to modify when updating to meet new requirements or correcting errors.

- **Interoperability**

  The stemmer project should be developed with an option to be integrated with other systems as well. This would enable external communications between the interacting systems and this could result in an increased scope, functions and operations for the system.

- **Understandability**

  Understandability in terms of the coding allows other programmers to understand the logic of the program flow. Thus, changes could be made easily upon the necessary program segments without modifying other essential logic of the program or restructuring the whole program itself.

- **Response time**

  Another aspect of this stemmer is its performance, which could be evaluated by the response time. The time taken by this stemmer to stem a word should not be too short (as this could result in inaccurate response) and should not be too long (as this could dissatisfy the system users).

3.6 Development Tools

The ultimate objective of a system development is to produce a functional system, which is as requested by the customer. In order to achieve this goal, choosing the right methods and methodologies is very important and could prove to be decisive at times. The same goes to the development tools. Choosing the appropriate tools would ease the burden of the developers and at the same time, fasten the system development process. The right development tools also aids in the process of developing a system in an orderly way. As for this Bahasa Melayu Stemmer, the development tools that have been used throughout the project are listed and described briefly in this section.

3.6.1 Hardware Requirements

The hardware requirements for this stemmer system development are:

- Intel Pentium IV 256 MHz Processor
- 128MB RAM
- 1GB Hard-Disk Space
- 16MB Video Card
- VGA Monitor
- HP690JC Printer
- Keyboard
- Mouse

3.6.2 Software Requirements

The hardware requirements for this stemmer system development are:

- **Microsoft Visual C++**

  The programming language that I have chosen to implement my stemmer is Microsoft Visual C++. I have opted for this programming language after a thorough and careful analysis on all the available programming languages. The C++ language was developed by Bjarne Stroustrup in the early 1980s as an enhancement of C language. The major characteristics of this programming language include use of objects, classes, constructors, destructors, inheritance, encapsulation and operator overloading. However, the power of C++ lies in its object-oriented method of programming. The manipulation of objects and classes in an object-oriented way ensures transparency and increases performance. Apart from that, data abstraction and 'inline' functions are supported by the C++ language as well and this is surely an added advantage.

- **Adobe Photoshop**

  There is another supporting software that I have to use in the development of my stemmer – Adobe Photoshop. I have chosen this software to edit the user interface of the stemmer project. Adobe Photoshop is one of the professional image-editing and graphic manipulation software. It provides functions such as scanning images, drawing backgrounds, producing photographs, creating graphics, performing colour correction, retouching and other image manipulations. Adobe Photoshop software introduces the next generation of

image editing with powerful new features which promotes efficiency and creativity. Delivering the broadest and most productive toolset available, Adobe Photoshop helps to achieve the highest quality results with faster performances.

3.7 Summary

This chapter explains the methodology used to develop my Bahasa Melayu Stemmer. The explanation of the chosen methodology, waterfall model, has been covered in great detail along with the justifications of why I had chosen the method. Apart from that, system requirements have been discussed as well. Functional and non-functional requirements of this proposed system have been presented and briefly explained. Another aspect that we have gone through is the development tools required to implement this stemmer project. This includes software and hardware requirements for the system development process. It is hoped that the chosen methods, methodologies and development tools are appropriate and well-suited for this system.

CHAPTER FOUR

SYSTEM ANALYSIS AND DESIGN

4.1 Introduction

A complete understanding of the proposed system is required before the development process could begin. System analysis and design is concerned with providing this type of understanding. System analysis is a problem-solving technique that decomposes a system into its components for the purpose of studying how well those components work and interact to accomplish their purpose. System design is a complementary problem-solving technique to system analysis that resembles a system's components back into a complete system. Together, system analysis and design can be referred to as a creative process of transforming the already defined requirements into system implementations. This chapter will discuss the system analysis and design process which is believed to be essential to the success of a system development.

4.2 System Model

A picture is worth a thousand words. A system model is worth a thousand explanations. No matter how much explanation you provide for a particular system, the explanation would be incomplete without pointing out a system model. Generally, system model is a representation of that system in reality. It paves the way for better understanding of that system and acts as a way to structure the unstructured problems

associated with the system. As for this stemmer project, I have chosen two types of models to represent the system, which are context model and data-flow model.

4.2.1 Context Model

Context model is an architectural model which illustrates the structure of the proposed system. It provides an high-level information about the system in development. The context model for this stemmer project is presented below:

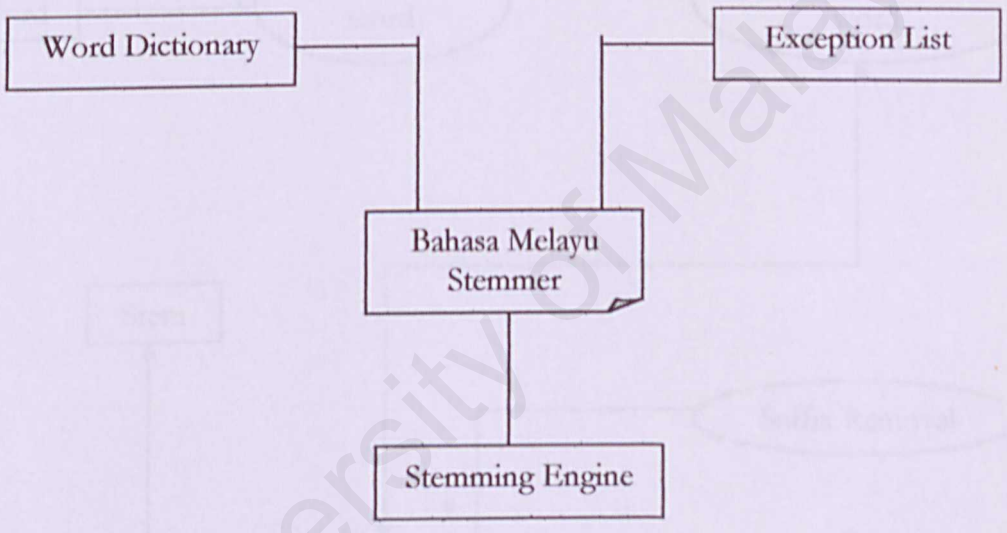


Figure 4.1 Context Diagram

4.2.2 Data-Flow Model

Data-flow model is a behavioural model which is used to describe the overall behaviour of the system. Data-flow model is an intuitive way of showing how data, or information particularly, are processed by a system. It provides a low-level, detailed

representation of the system in development. The data-flow model for this stemmer project is presented below:

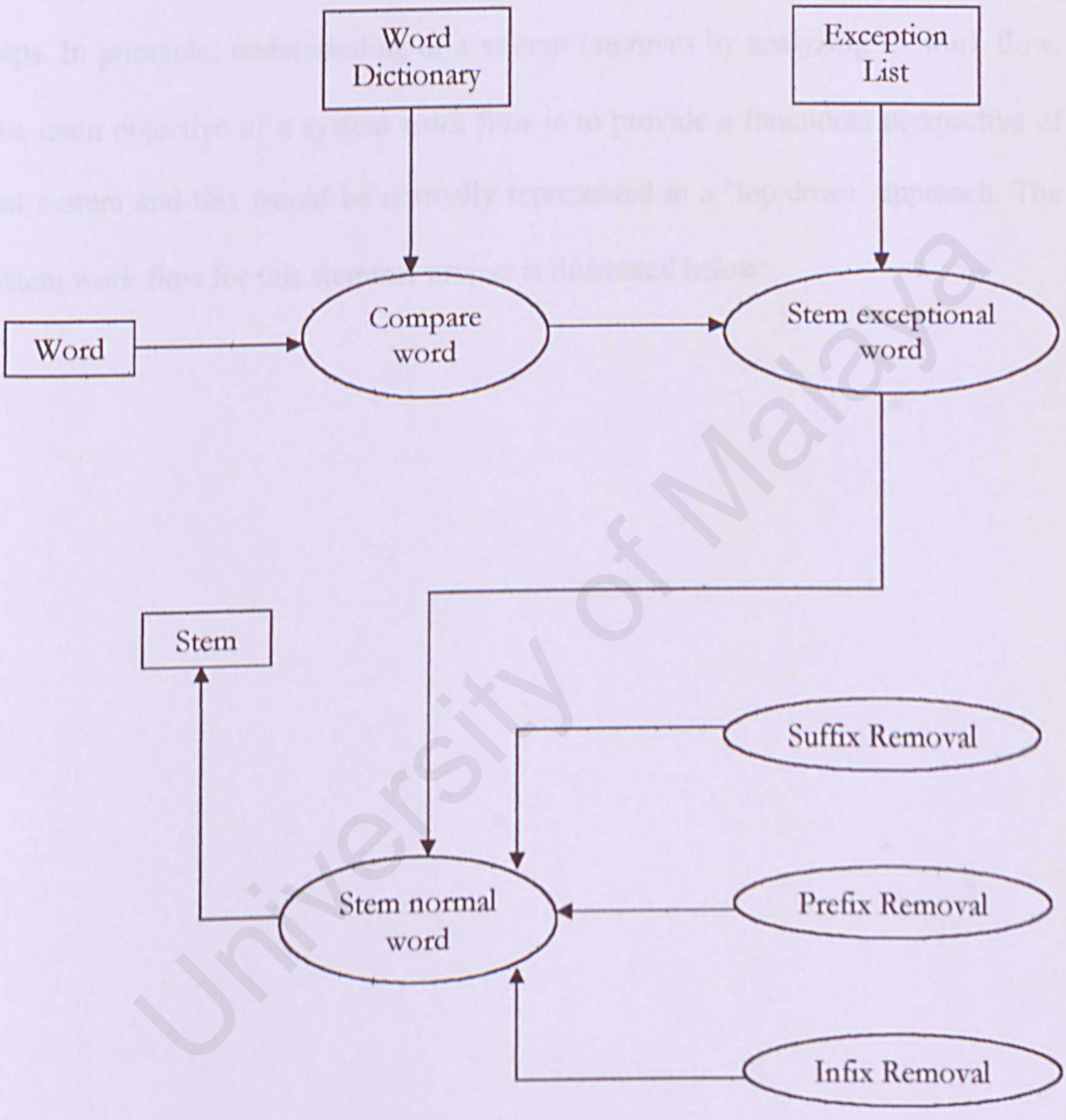


Figure 4.2 Data-Flow Diagram

4.3 System Work-Flow

Generally, system work flow process provides a detailed description of how a system is designed and how does its components interact with each other to work as a system. It is used to represent the flowing of data through a sequence of processing steps. In principle, understanding of a system improves by analyzing its work flow. The main objective of a system work flow is to provide a functional perspective of that system and this would be normally represented in a 'top-down' approach. The system work flow for this stemmer project is illustrated below:

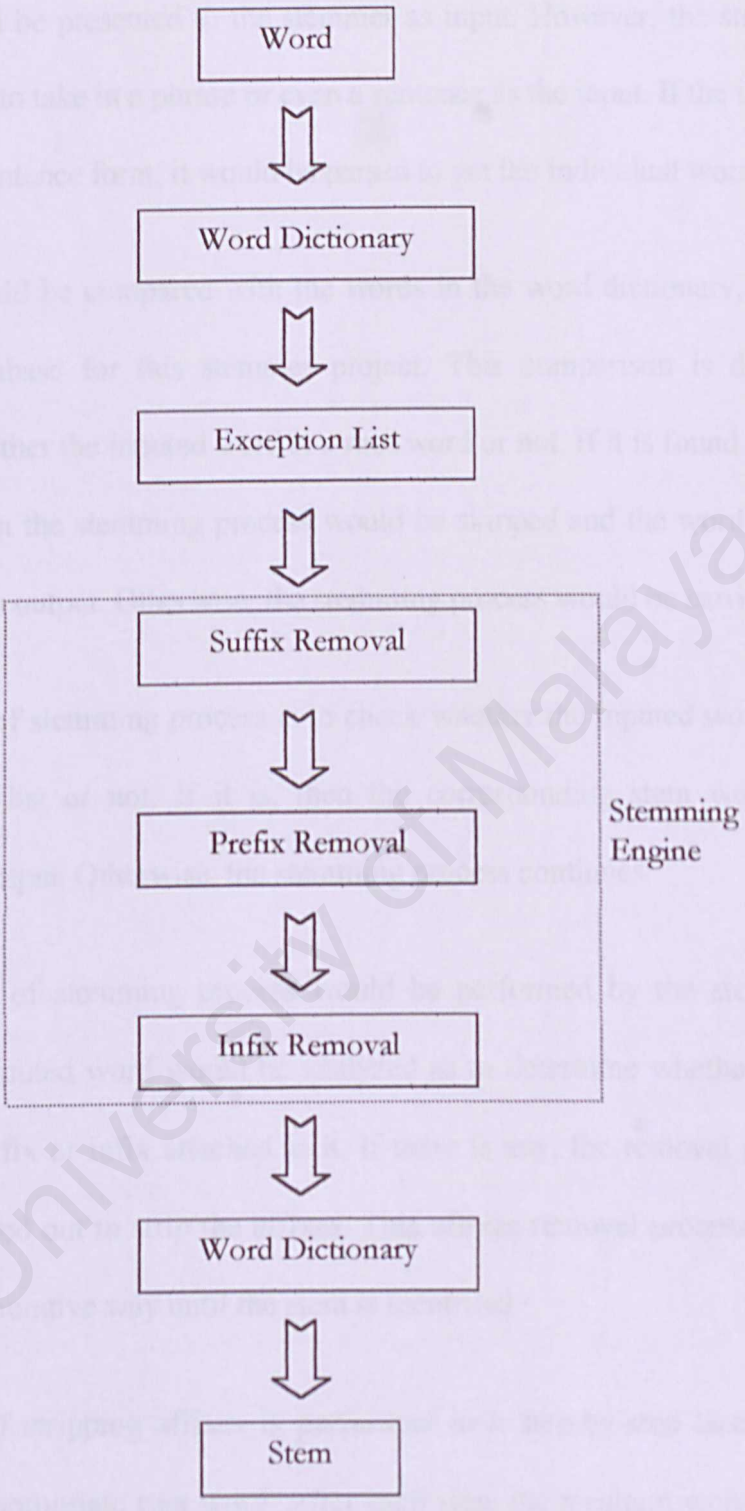


Figure 4.3 System Work-Flow Diagram

Explanation

- A word would be presented to the stemmer as input. However, the stemmer would be able to take in a phrase or even a sentence as the input. If the input is in phrase or sentence form, it would be parsed to get the individual words

- The word would be compared with the words in the word dictionary, which acts as a database for this stemmer project. This comparison is done to determine whether the inputed word is a root word or not. If it is found to be a root word, then the stemming process would be skipped and the word would be presented as output. Otherwise, the stemming process would be carried out

- The first step of stemming process is to check whether the inputed word is in the exception list or not. If it is, then the corresponding stem would be presented as output. Otherwise, the stemming process continues

- The next step of stemming process would be performed by the stemming engine. The inputed word would be analyzed as to determine whether it has any suffix, prefix or infix attached to it. If there is any, the removal process would be carried out to strip the affixes. This affixes removal process would be done in an iterative way until the stem is identified

- The process of stripping affixes is performed in a step-by-step basis as to generate the appropriate root word. After each step, the resultant word would be compared with the list in the word dictionary. This is done to ensure that the stemmer does not overstem a given word

- When the stemming process has been completed, the inputed word has been reduced to its root form and the resultant word would be presented as stem, the final output of this stemmer

4.4 Interface Design

Computer systems are meant for users and it is developed exclusively for them. Without users, a system would not serve its purpose. Thus, users are such an essential element of system development. So does user interface. User interface plays a very important role in determining the quality of a system. An interface that is difficult to use will, at best, result in a high level of user errors. At worst, users will simply refuse to use the system irrespective of its functionality. Therefore, good user interface design is critical to the success of a system development.

System data, both input and output, are most dependent on a well-designed user interface. Basically, an interface design should meet the expected attributes such as objectiveness, consistency, simplicity, accuracy, attractiveness and ease of use. This attributes are very much a necessity for a good user interface design as to avoid misplacing of information. If information is presented in a confusing or misleading way, users may misunderstand the meaning of that particular information.

As to avoid and overcome all these problems, a few guidelines have been considered in designing the user interface for this stemmer system. The following aspects or guidelines have been carefully considered:

- Only relevant information should be displayed
- Presentation format used should be easy to understand
- Standard abbreviations and consistent labels should be used
- Meaningful and appropriate error messages should be displayed
- Consistency should be maintained in overall interface design issues

The proposed user interface design for this project is as follows:

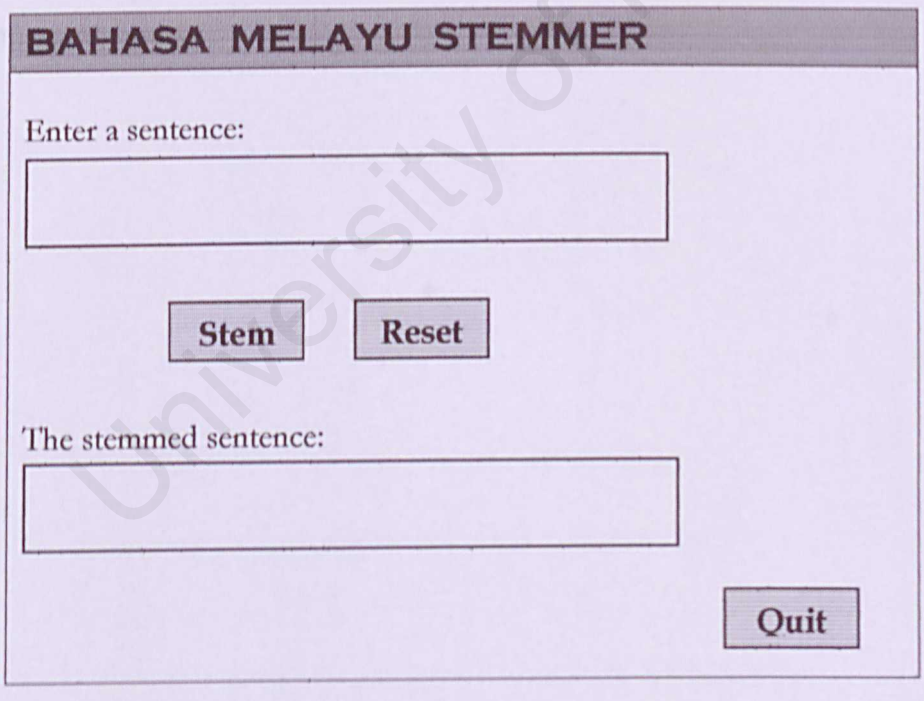


Figure 4.4 User Interface Design

4.5 Summary

This chapter explained the analysis and design process which have been performed on this Bahasa Melayu Stemmer. System models have been presented to provide better understanding of the system. Two system models have been explored; context model and data-flow model. In addition, system work flow has been illustrated as well with special focus given to the flow of the system in term of its sequential processes. This is hoped to further enhances one's understanding of this stemmer project. Another design issue which has been discussed in this chapter is the user interface design. This design issue has been given utmost importance since users are such an essential element in the development of a system. Appropriate and consistent designs have been applied and adapted throughout this development process as to relish the dream of system completion.

CHAPTER FIVE

SYSTEM IMPLEMENTATION

5.1 Introduction

System implementation is well-defined as the integration of the physical and conceptual resources that produce a working system. Therefore, system implementation is the physical realization of a software development process. In other words, system implementation can be thought of a process which converts a system specification into an executable system.

System implementation is the delivery of a system into its production, meaning day-to-day operation (*Whitten 2000*). Normally, this implementation phase in a system development process would involve the work of programming, but not necessarily. Refinement of the software design specification could also take place in this development stage as well.

Among the major steps involved in system implementation are the construction of the system, installation of the constructed system and conversion of the system. The purpose of system construction is basically to develop a fully functional system that fulfills the design requirements. As for this *Bahasa Melayu Stemmer*, the construction phase would involve database creation and also the system coding. Coding, or normally referred to as programming, is generally recognized as a major aspect of the system construction.

Installation of the constructed system is done to integrate all the functions in each module being built for the system. System conversion provides a smooth transition from the development stage to the operational stage; and help users to cope with normal start-up problems. Therefore, the combination of system installation and conversion would produce a deliverable of an operational system that will function as required by the objectives and specifications of the system.

5.2 Stemming Algorithm

As mentioned in the preceding section, there are two major parts involved in the implementation phase for this stemmer; namely database creation and system coding. However, before getting down to the actual implementation part, there is something to figure out beforehand – stemming algorithm. Stemming algorithm is basically an important aspect in this system development as a stemmer could not be successfully implemented without a proper stemming algorithm. A general algorithm has been presented in the previous chapter and in this section, a more specific and detailed stemming algorithm will be exemplified and this is as follows:

Step 1

Check input word against the word dictionary

If the word is found in the dictionary, then

Output the word as root word

Else

Go to Step 2

Step 2

Check input word for any prefix

If the word has a prefix, then

Remove the prefix and go to Step 1

Else

Go to Step 3

Step 3

Check input word for any suffix

If the word has a suffix, then

Remove the suffix and go to Step 1

Else

Go to Step 4

Step 4

Check input word for any infix

If the word has a infix, then

Remove the infix and go to Step 1

Else

Go to Step 5

Step 5

Check stemmed word for its first letter

If the first letter starts with 'm', then

Replace the first letter with 'p' or 'f' and go to Step 1

If the first letter starts with 'n', then

Replace the first letter with 't' and go to Step 1

If the first letter starts with 'y', then

Replace the first letter with 's' and go to Step 1

If the first letter starts with a vowel, then

Add 'k' as the first letter and go to Step 1

Else

Root word could not be found and the program terminates

5.3 Database Creation

The first step to any system implementation is to create the system database. The same goes to my project, *Bahasa Melayu Stemmer*. As mentioned in the previous chapter, this stemmer would require a database of Malay words in order to function appropriately. Thus, the preparatory point for this stemmer project would be the process of database creation.

Preparing a database for this stemmer is very much dissimilar to that of creating databases for other projects or programs. Unlike other systems which use databases such as MSAccess or MySQL, the database for this *Bahasa Melayu Stemmer* is quite simple and straightforward. It is a text (.txt) file which contains approximately 12,000 Malay words; just root words.

All the words are resourced from the premier Malay dictionary, *Kamus Dewan*, from the well-known publisher of *Dewan Bahasa dan Pustaka*. The words are well organized in an alphabetical order; each word in this database is typed in a single line to accommodate for easier and faster file processing. The database is then named as '*bma-z.txt*' (abbreviation for 'bahasa melayu a-z words') and is saved in the same folder (or location) as the stemmer project.

A sample of the database (text file dictionary) is as follows:

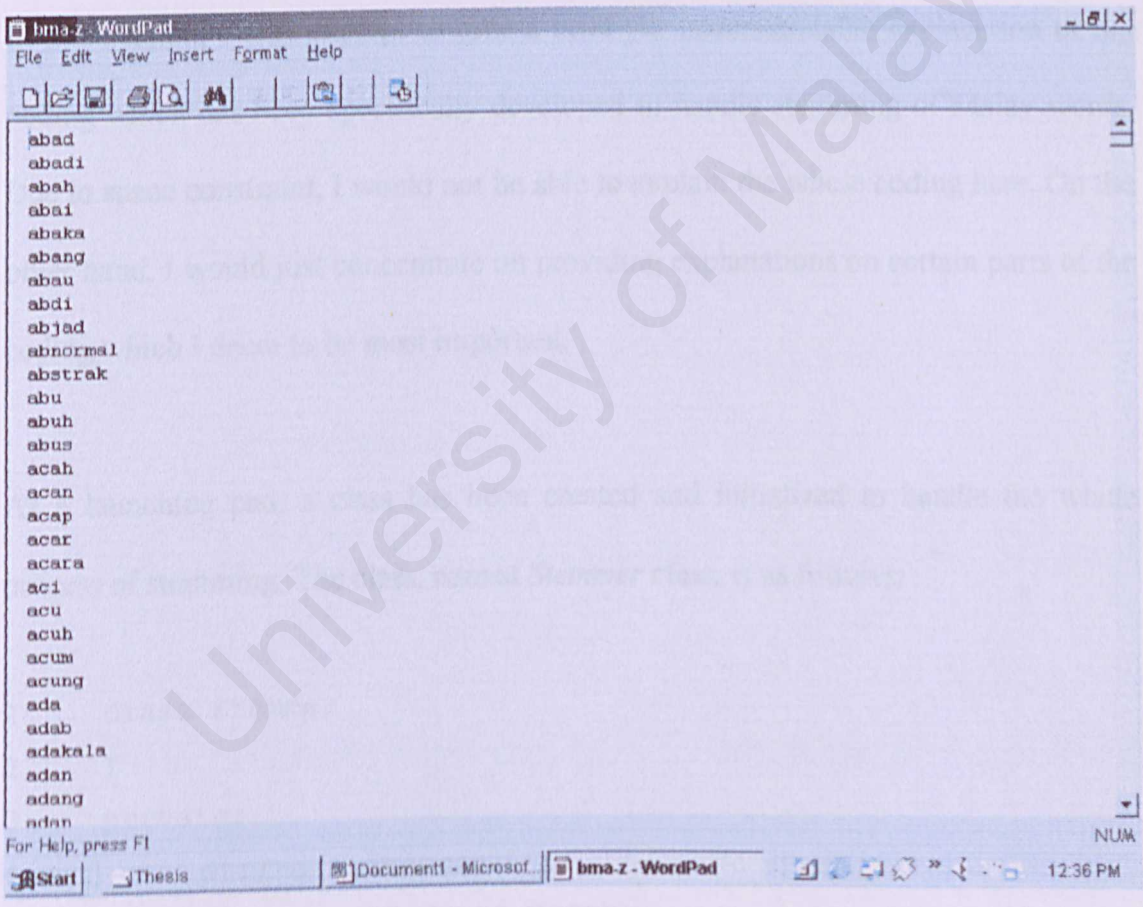


Figure 5.1 Sample of Database (Text File Dictionary)

5.4 System Coding

Once the database development has been completed, we will begin with the system development phase. We will start off the programming work by developing a fully functional or complete coding for this stemmer system. Coding normally involves steps that translate a detailed design representation of software into a program language realization. As for my *Bahasa Melayu Stemmer*, I have chosen Microsoft Visual C++ as my programming tool.

In this section, I will attempt to give a brief yet understandable explanation of my coding which has been specifically developed to handle stemming of Malay words. Due to space constraint, I would not be able to explain the whole coding here. On the other hand, I would just concentrate on providing explanations on certain parts of the coding which I deem to be most important.

As a launching pad, a class has been created and initialized to handle the whole process of stemming. The class, named *Stemmer* class, is as follows:

```
1    class Stemmer
2    {
3    private:
4         string sentence;
5         string word[NO_OF_WORDS];
6         int flag[NO_OF_WORDS];
7    public:
8         Stemmer();
9         void setFlag(int);
```

```
10          void setSentence(string);
11          void setWord(string, int);
12          void stem(string, int);
13          void stemExtra(int);
14          void stemDouble(int);
15          void stemInfix(int);
16          void stemException(int);
17          bool check(string);
18          bool isVowel(char);
19          int getFlag(int);
20          string getSentence();
21          string getWord(int);
22      };
```

Now, let's navigate through each line of the class presented above.

- Line 1 defines the name of the class as '*Stemmer*'.
- Line 3 delineates the private data members for the '*Stemmer*' class. There are three private data members declared here.
- The variable '*string sentence*' in line 4 is defined to hold the inputted sentence – the whole sentence.
- The variable '*string word*' in line 5 is defined as arrays to hold a word each from the inputted sentence.
- The last variable, '*int flag*' in line 6 acts just like a 'flag'. It is used to determine whether a word has been stemmed or not. When a word has

been stemmed, the '*int flag*' variable would be '*flagged*' and this determines the end of stemming process for that particular word.

- Line 7 delineates the member functions (methods) for the '*Stemmer*' class. There are a total of 14 methods declared here.

- '*Stemmer()*' in line 8 is the constructor for this '*Stemmer*' class. It acts to initialize the private data members with default values.

- The next three methods, '*void setFlag()*', '*void setSentence()*' and '*void setWord()*' initializes the corresponding private data members, '*int flag*', '*string sentence*' and '*string word*'.

- The method '*void stem()*' in line 12 is defined to handle the process of basic stemming, both prefixes and suffixes. Other parts of stemming, which are a bit harder, are handled by other methods.

- The method '*void stemExtra()*' in line 13 is defined to handle some special suffixes, which is not covered by the previous method.

- The method '*void stemDouble()*' in line 14 is defined to handle 'kata ganda' (double words). This method would be able to stem the double word to its equivalent stem or root word.

- The method '*void stemInfix()*' in line 15 is defined to handle words with infixes. This type of words are processed later on because the

number of Malay words with infixes is very rare – this could improve the performance of this stemmer.

- The method '*void stemException()*' in line 16 is defined to handle words with exception cases. As not all the words follow a standard stemming rule or algorithm, this method is vital to stem affixes from this so-called special circumstanced words.

- Line 17 defines a method called '*bool check()*'. This method does the comparing process, where it compares the stemmed word with the root words in the database. If both the word matches, it returns the value of 'true'; and 'false' if vice versa.

- Line 18 defines a method called '*bool isVowel()*'. This method checks whether a character is vowel or a consonant. If it is a vowel, this method returns the truth value; and vice versa.

- The last three methods, '*int getFlag()*', '*string getSentence()*' and '*string getWord()*' returns the value contained in the corresponding private data members, '*int flag*', '*string sentence*' and '*string word*'.

We have gone through the declaration of the '*Stemmer*' class and its associated private data members as well as its member functions (methods). As for the definition part of the '*Stemmer*' class, presenting the whole definition of the class would be tedious and time-consuming. Therefore, we would just consider a few important examples which outlines the basic algorithm of stemming.

Consider this function which stems the prefix '*ber*':

```
1     if((s[0] == 'b') && (s[1] == 'e') && (s[2] == 'r'))
2     {
3           string temp(s);
4
5           //remove be***
6           if(check(temp.substr(2,string::npos)))
7           {
8                 s.erase(0,2);
9                 setWord(s,i);
10                setFlag(i);
11          }
12
13          //remove ber***
14          else
15          {
16                s.erase(0,3);
17                setWord(s,i);
18                setFlag(i);
19          }
20    }
```

In the above function,

- Line 6 to 11 tries to remove the prefix '*be*'. For example,

  ***be**rasa* → *rasa*

- Line 14 to 19 tries to remove the prefix '*ber*'. For example,

  ***ber**lari* → *lari*

Consider this function which stems the suffix '*kan*':

```
1     //calculate length of word
2     int d = s.length() - 1;
3
4     if((s[d-2] == 'k') && (s[d-1] == 'a') && (s[d] == 'n'))
5     {
6           string temp(s);
7
8           //remove ***an
9           if(check(temp.substr(0,d-1)))
10          {
11                s.erase(d-1,2);
12                setWord(s,i);
13                setFlag(i);
14          }
15
16          //remove ***kan
17          else
18          {
19                s.erase(d-2,3);
20                setWord(s,i);
21                setFlag(i);
22          }
23    }
```

In the above function,

- Line 9 to 14 tries to remove the suffix '*an*'. For example,

  *semak**an*** → *semak*

- Line 17 to 22 tries to remove the suffix '*kan*'. For example,

  *jalan**kan*** → *jalan*

Consider this function which stems the infix '*el*':

```
1     if((s[1] == 'e') && (s[2] == 'l'))
2     {
3           string temp(s);
4           temp.erase(1,2);
5
6           //remove infix
7           if(check(temp))
8           {
9                 s.erase(1,2);
10                setWord(s,i);
11                setFlag(i);
12          }
13    }
```

In the above function,

- Line 7 to 12 tries to remove the infix '*el*'. For example,

  *t**el**apak* → *tapak*

Consider this function which stems the double words ('kata ganda'):

```
1     if(s.find("-") != string::npos)
2     {
3           int d = s.length() - 1;
4
5           int x = s.find("-");
6
7           //remove double words ('kata ganda')
8           if(check(s.substr(0,x)))
9           {
10                s.erase(x);
11                setWord(s,i);
12                setFlag(i);
13          }
14
15          else if(check(s.substr(2,x-2)))
16          {
17                s.erase(x);
18                s.erase(0,2);
19                setWord(s,i);
20                setFlag(i);
21          }
22
23          else if(check(s.substr(3,x-3)))
24          {
25                s.erase(x);
26                s.erase(0,3);
27                setWord(s,i);
28                setFlag(i);
29          }
30
```

```
31          else if(check(s.substr(4,x-4)))
32          {
33                  s.erase(x);
34                  s.erase(0,4);
35                  setWord(s,i);
36                  setFlag(i);
37          }
38      }
```

In the above function,

- Line 8 to 13 tries to remove double words ('kata ganda') with no prefixes. For example,

  kapal-kapal → *kapal*

- Line 15 to 21 tries to remove double words ('kata ganda') with two prefixes. For example,

  se*akan-akan* → *akan*

- Line 23 to 29 tries to remove double words ('kata ganda') with three prefixes. For example,

  ber*lumba-lumba* → *lumba*

- Line 31 to 37 tries to remove double words ('kata ganda') with four prefixes. For example,

  meng*ayun-ayun* → *ayun*

The above examples have outlined four algorithms for basic stemming in the form of programming language realization. The first two algorithms, the stemming of prefixes and suffixes, are more or less the same for other types of prefixes and suffixes. However, the last two algorithms, the stemming of infixes and the double words are exactly the same for other types of infixes and double words.

5.5 Coding Methodology

The coding methodology undertaken in the development of this *Bahasa Melayu Stemmer* is the top-down methodology, which is also called as the divide and conquer method. The top-down methodology can be described as a development approach, in which development of the simple sub-modules (higher level modules) are carried out first and then followed by the more complex sub-modules (lower level modules).

The purpose of using the top-down approach is to enable testing processes to be done on the simple sub-modules while the complex sub-modules are still in the process of coding. By applying this approach, the testing procedures could be carried out even though that the system has not been completed. A part of system testing could overlap the development process and this would ensure the faster development and completion of a system.

Besides that, developing the simple sub-modules first would enable the checking process as well. The flow of the system could be checked as to whether it is similar to

the proposed design or not; and the connectivity of each module could be seen clearly. Another advantage of using this methodology is to ensure that the most important modules to be created and tested first. It also gives a preliminary outline of how the system will work before it is completed.

5.6 Coding Style

The coding style is a very important attribute that determines the level of a coding. It proves decisive in establishing the readability and maintainability of the source codes, which is coding after all. With a systematic and apparent coding style, it helps the programmer to read the codes clearly and easily. The source codes would be more understandable for the programmer to maintain and debug. This would ensure that future modifications could be done to the system coding whenever needed and without any problem or hassle.

In general, there are many rules of thumbs in defining a good coding practice. For this stemmer project, a lot of the precaution steps have been taken and implemented during the coding process. It is hoped that all these steps would improve the goal of the coding itself – to make it more readable and understandable. A good coding practice will not only enhance readability but it is also a crucial step for future maintenance and modification.

Several coding practices have been employed throughout the program or source codes for this stemmer system. These practices will ultimately determine the coding style of this *Bahasa Melayu Stemmer* in order to ensure system readability, maintainability and consistency. The followings show some of the coding practices and styles applied for this *Bahasa Melayu Stemmer*:

Meaningful Variable Names

- Choosing the right and meaningful variable names can help to reflect their usage and purpose in the source codes. For example, in the coding for this system, the variable '*sentence*' is used to store a sentence while the words of that sentence are held by the variable '*word*'.

- Choosing the meaningful names for variables also helps in the testing process. It helps one to easily trace flaws in the source codes and at the same time, to enable references to be done easily.

- Meaningful variable names also serve a purpose for program or source codes to be 'self-documenting' without excessive use of comments.

Formatting to Enhance Understanding

- Formats do make a program more readable and understandable, provided that the formatting is done in a standard form and this form is maintained throughout the program.

- The format of statements such as proper indentation can be applied to reflect the basic control structures of the coding.

- Spacing of statements is another way of applying some kind of format to source codes. For example, the methods used could be separated by a blank line to indicate the different types of definitions.

5.7 Summary

This chapter described the system implementation tasks which had been carried out in order to turn the dream of building a stemmer system into a reality. The first and foremost step for constructing a system is to create its database. As for this *Bahasa Melayu Stemmer*, a database (text file dictionary) has been created to store all the Malay root words. When this has been completed, the process of system coding continues. System coding, or programming, is essentially known as the most important aspect of system implementation. A few samples of the source codes which are written for this stemmer project were presented and briefly explained. Coding methodology was also presented thereafter, in which the approaches chosen in developing this stemmer system were explained. Some good programming or coding styles were applied throughout the system implementation process and this was explained as well in the concluding part.

CHAPTER SIX

SYSTEM TESTING

6.1 Introduction

System testing, as the name suggests, is all about testing a constructed system. System testing involves executing an implementation of the system with test data and examining the outputs of the system and its operational behaviour to check that it is performing as required. Testing is a dynamic activity because it just works with an executable representation of the system.

System testing also involves exercising the system using real-like data to detect defects and flaws. The existence of program defects or inadequacies is inferred by examining the outputs of the program and looking for anomalies. These defects are then fixed or corrected accordingly as to produce an improved and less erroneous system. The testing process may be carried out during the implementation phase and also after the implementation phase has been completed.

However, this process of system testing is actually a process of verification and validation of a developed system. Verification and validation are closely associated with checking and analysis processes that ensure the system conforms to its goals and specifications. Verification and validation are not the same thing although they are easily confused to be (*Boehm 1989*).

The difference between these two is succinctly expressed as follows:

- Validation answers the question 'are we building the right product?'
- Verification answers the question 'are we building the product right?'

These definitions details out the role of verification as checking that the system conforms to its specification. On the other hand, validation goes beyond checking conformance of the system to its specification to showing that the system does what the user or customer expects as distinct from what has been specified. Therefore, system testing which makes use of both validation and verification, will ensure that the developed system will function as required.

The main purpose of system testing is to uncover defects and deficiencies that exist while implementing the system. A successful testing process will cause the system to perform incorrectly and hence exposes a defect. This emphasizes an important fact about testing – testing demonstrates the presence, not the absence, of program faults. As such, system testing is a critical element of software quality assurance and represents the ultimate review of specification and implementation.

6.2 Testing Strategy

Testing strategy reflects an approach to system testing – it determines how the testing process is carried out on the constructed system. As for this *Bahasa Melayu Stemmer*, the testing strategy chosen is the bottom-up strategy. Since the top-down approach has been chosen for the coding methodology, it has to be its opposite form in the

testing process as to avoid any discrepancies. This would ensure a complete and comprehensive testing process to be carried out on every part of the system. This prompts the choice of bottom-up testing strategy after all.

Bottom-up testing is an integral part of a bottom-up development process where the development process starts with low-level components and works up the component hierarchy. As such, the bottom-up testing strategy works in the same way – by integrating and testing the low-level components of the system before moving upwards to test the high-level components.

After the low-level component has been programmed and tested, its higher-level components are implemented and tested in the same way. This process continues until the top-level components are implemented and tested. The whole system has then been completely tested. This would ensure that flaws and defects associated with the system has been detected and corrected accordingly.

6.3 Testing Manual

Testing should be planned before implemented. Careful planning is needed to get the most out of the testing process and to ensure transparency is maintained throughout. Test planning is concerned with setting out standards for the testing process rather than describing the testing process itself. The planning process should also draw up

standards and procedures for system testing, establish checklists to drive system testing and define the testing manual.

A testing manual, or a system test plan, is an essential document that assists in the system testing process. An absolute and comprehensive testing manual is needed to ensure the effectiveness of the testing process. Given the importance of the testing manual, it has been a top priority to come up with a complete test plan for this stemmer project, which has been more or less realized.

The test plan for this *Bahasa Melayu Stemmer* is based on the List of Malay Affixes. Presenting the whole test plan in this section would be less useful and redundant after all. Thus, it is reserved for the next section, where the complete test plan are presented together with the testing results as this could enhance one's understanding about the overall testing process.

6.4 Types of Testing

There are various types of testing procedures available to ensure completeness and correctness of a developed system. In general, systems should not be tested as a single, monolithic unit as this would defeat the purpose of system testing itself. As most of the software systems are large systems which are built out of many sub-systems, modules, procedures and functions, the testing process should therefore proceed incrementally in stages.

The following figure shows a four-stage testing process which have been adapted to this *Bahasa Melayu Stemmer*:

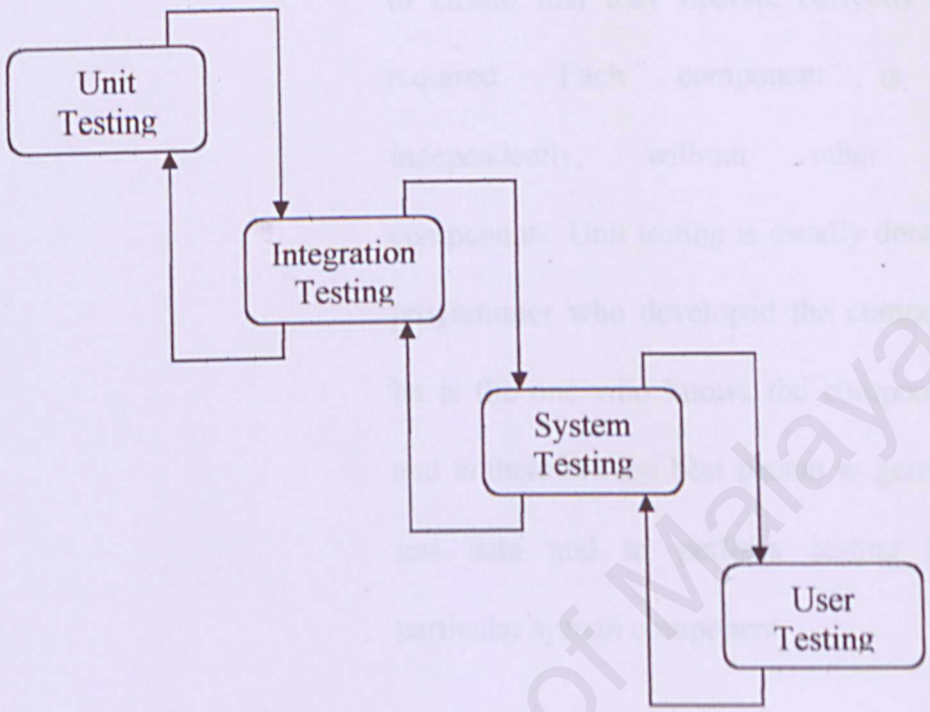


Figure 6.1 Testing Process Stages

The above figure indicates that system components are tested first, followed by testing the integrated system and finally, the whole system is tested. Ideally, component defects are discovered early in the process while integration problems arise when the system is integrated. However, as defects are discovered, the program must be debugged and this may require other stages in the testing process to be repeated. Errors in program components, say, may come to light during integration testing. The process is therefore an iterative one with information being fed back from later stages to earlier parts of the process.

The four stages in the testing process are as follows:

- Unit Testing - Individual components of the system are tested to ensure that they operate correctly and as required. Each component is tested independently, without other system components. Unit testing is usually done by the programmer who developed the component as he is the one who knows the component best and is therefore the best person to generate the test data and to perform testing for that particular system component.

- Integration Testing - This phase involves testing collections of the system units which have been integrated into sub-systems. Thus, this integration testing is also normally known as sub-system testing. A larger part of the system is integrated and tested to ensure that it conforms to its specification and expectations. This normally involves integrating work from a number of programmers or an independent team of testers.

- System Testing - The sub-systems are integrated to make up the system. This process is concerned with finding errors that result from unanticipated interactions between sub-systems. It is also concerned with validating that the system meets its functional and non-functional requirements and testing the emergent system properties.

- User Testing - This is the ultimate stage in the testing process before the system is accepted for operational use. The system is tested with data supplied by the users rather than using stimulated data. User testing may reveal errors and omissions which have not been experienced so far because the real data exercise the system in different ways from the test data.

6.5 Results of Testing

Different types of testing have been explained in the previous section and all of these testing types will be performed on this stemmer system. This section will outline the testing processes which have been carried out throughout the implementation phases of this system. Results of the test will be presented and any flaws or defects arising

from this testing procedure would be fixed accordingly. Now, let's go through to each type of testing and see how well this system fares.

6.5.1 Unit Testing

In this stemmer system, each of the basic stemming functions are implemented as an unit. Thus, testing each one of these functions independently would be identical to that of performing unit testing. Among the functions that would be tested as an unit are the prefix stemming function, suffix stemming function, infix stemming function, dual word ('kata ganda') stemming function and root word checking function.

Root Word Checking Function

As a way to start off the testing process, this system would be tested with root words (the system should return back the root word as it is and should not perform any stemming process). The results of this testing are as follows:

Test Data (Root Word)	Stem	Ind
bintang	bintang	✓
kekacang	kekacang	✓
nasionalisme	nasionalisme	✓
perikemanusiaan	perikemanusiaan	✓
zuriat	zuriat	✓

Table 6.1 Test Data and Result for Root Word Checking

where,

Test Data represents the words that were used for testing

Stem indicates the corresponding output (test result)

Ind (Indication) shows whether the intended output has been achieved (the symbol ✓ denotes a success while the symbol ✗ denotes a failure)

The above table shows that the root word checking function is working properly.

Prefix Stemming Function

The next function to be tested is the prefix stemming function. Words with attached prefix would be inputted to this stemmer system and it should be able to identify and remove the prefix; and return back the root word as its output. The results of this testing are as follows:

Test Data (Prefix)	Word	Stem1	Stem2	Ind
Words starting with '*ber*'	**ber**kelip	kelip	kelip	✓
Words starting with '*per*'	**per**buat	buat	buat	✓
Words starting with '*ter*'	**ter**lantar	lantar	lantar	✓
Words starting with '*mem*'	**mem**beri	beri	beri	✓
Words starting with '*pem*'	**pem**bawa	bawa	bawa	✓
Words starting with '*menge*'	**menge**pos	pos	pos	✓
Words starting with '*penge*'	**penge**bom	bom	bom	✓
Words starting with '*meng*'	**meng**hapus	hapus	hapus	✓
Words starting with '*peng*'	**peng**awas	awas	awas	✓
Words starting with '*men*'	**men**cari	cari	cari	✓
Words starting with '*pen*'	**pen**dakwa	dakwa	dakwa	✓
Words starting with '*me*'	**me**masak	masak	asak	×
Words starting with '*pe*'	**pe**kerja	kerja	kerja	✓
Words starting with '*be*'	**be**rehat	rehat	berehat	×
Words starting with '*ke*'	**ke**muka	muka	muka	✓
Words starting with '*se*'	**se**punya	punya	sepunya	×
Words starting with '*te*'	**te**rasa	rasa	asa	×
Words starting with '*di*'	**di**didik	didik	didik	✓

Table 6.2 Test Data and Result for Prefix Stemming

where,

Test Data represents the type of words that were used for testing

Word represents the words that were used for testing

Stem1 indicates the intended output (target result)

Stem2 indicates the corresponding output (test result)

Ind (Indication) shows whether the intended output has been achieved
(the symbol ✓ denotes a success while the symbol ✗ denotes a failure)
(*please note that all these explanations applies to the following tables as well*)

The above table demonstrates that the prefix stemming function works well for most of the tested cases. However, there are a few flaws that were detected:

1. This stemmer is unable to stem the prefix '*me*'. This is down to the fact that the stemming function could not differentiate the words starting with the prefix '*me*' (the root word starts from the letter '*m*') with the words starting with the prefix '*mem*'. As such, the word '*memasak*' is confused to return the root word of '*asak*' instead of '*masak*'.

   Correction

   For the words that begin with the string '*mem*', the word is checked first whether the prefix is really '*mem*' or '*me*'. When this ambiguity has been resolved, only then the stemming process is performed. By implementing this checking method, the function could stem the word according to the right prefix, thus returning the right root word.

2. The system is unable to stem the prefix '*be*'. This is again due to the fact that the words starting with the prefix '*be*' (the root word starts from the letter '*r*') are bemused with the words starting with the prefix '*ber*'. As such, the word '*berehat*' is returned without being stemmed as the word '*ehat*' does not exist in the Malay language.

Correction

The correction method that has been used to solve the first defect is reapplied here. Words that begin with the string '*ber*' are checked whether the real prefix is '*ber*' or '*be*'. The vagueness is resolved before continuing with the stemming process as to ascertain its accuracy.

3. The third defect is somehow a bit different. The stemmer tries to remove the prefix '*se*' from the word '*sepunya*' but it is unable to do it since the ending string '*nya*' is thought to be a suffix and therefore removed from that word. As such, the system could not find any word such as '*sepu*' in the database so the original word is returned as its output.

Correction

The removal of the suffix '*nya*' is modified as to craft it into a controlled stemming method. The suffix '*nya*' is only removed if the preceding string of that word resembles to a stem or root word. On the other hand, the removal of the suffix '*nya*' is skipped and other prefixes are tested. This would enable the identification of the prefix '*se*' from the word '*sepunya*' and its subsequent removal as to produce the accurate root word.

4. The last defect, the inability to stem the prefix '*te*' is again associated with the problem of mismatching the words starting with the prefix '*te*' (the root word starts from the letter '*r*') with the words starting with the prefix '*ter*'. Thus, the word '*asa*' is returned as the root word for the word '*terasa*'.

Correction

The checking step is again included in this prefix stemming element to solve the problem. Words beginning from the string '*ter*' are verified whether the real prefix is '*ter*' or '*te*'. Stemming process continues thereafter.

Suffix Stemming Function

The testing process would be continued then with the suffix stemming function. This stemmer system would take in words with attached suffix and it will try to identify and remove that suffix; and return back the root word as its output. The results of this testing are as follows:

Test Data (Suffix)	Word	Stem1	Stem2	Ind
Words ending with '*nya*'	bangga**nya**	bangga	bangga	✓
Words ending with '*kan*'	jalan**kan**	jalan	jalan	✓
Words ending with '*an*'	minum**an**	minum	minum	✓
Words ending with '*i*'	akhir**i**	akhir	akhir	✓
Words ending with '*kah*'	apa**kah**	apa	apa	✓
Words ending with '*lah*'	itu**lah**	itu	itu	✓
Words ending with '*pun*'	telah**pun**	telah	telah	✓
Words ending with '*ita*'	biduan**ita**	biduan	biduan	✓
Words ending with '*man*'	budi**man**	budi	budi	✓
Words ending with '*wan*'	harta**wan**	harta	harta	✓
Words ending with '*wati*'	seni**wati**	seni	seni	✓
Words ending with '*ku*'	sayang**ku**	sayang	sayang	✓
Words ending with '*mu*'	untuk**mu**	untuk	untuk	✓

Table 6.3 Test Data and Result for Suffix Stemming

The above table indicates that the suffix stemming function is working correctly.

'Kata Ganda' Stemming Function

Testing procedure is continued with the dual word ('kata ganda') stemming function. This system should be able to identify dual words in the first place, and then to remove any affixes (both prefixes and suffixes) attached to that particular word. The results of this testing are as follows:

Test Data ('Kata Ganda')	Word	Stem1	Stem2	Ind
Words without a prefix (identical words)	kapal-kapal	kapal	kapal	✓
Words without a prefix (non-identical words)	bolak-balik	bolak	bolak	✓
Words without a prefix (non-identical words)	saudara-mara	saudara	saudara	✓
Words with a prefix but without a suffix	**ber**lari-lari	lari	lari	✓
Words with a prefix but without a suffix	**ter**tanya-tanya	tanya	tanya	✓
Words with a prefix but without a suffix	**mem**beli-belah	beli	beli	✓
Words with a prefix but without a suffix	**men**deru-deru	deru	deru	✓
Words with a prefix but without a suffix	**meng**elak-elak	elak	elak	✓
Words with a prefix but without a suffix	**me**lihat-lihat	lihat	lihat	✓
Words with a prefix but without a suffix	**se**akan-akan	akan	akan	✓
Words with a suffix but without a prefix	satu-satu**nya**	satu	satu	✓
Words with a prefix and with a suffix	**ke**anak-anak**an**	anak	anak	✓
Words with a prefix and with a suffix	**se**baik-baik**nya**	baik	baik	✓

Table 6.4 Test Data and Result for 'Kata Ganda' Stemming

The above table demonstrates that the dual word ('kata ganda') stemming function is working perfectly and as intended.

6.5.2 Integration Testing

Each unit has been tested independently at this point and the detected defects have been rectified or resolved. The next testing phase would be to test the integrated components of this stemmer system. There are basically just two parts that make up this integration testing; namely prefix stemming function and suffix stemming function, which could be integrated in two different ways – integration of two affixes and integration of more than two affixes.

Integration of Two Affixes

In this section, the two most important stemming functions (prefix stemming function and suffix stemming function) are integrated as to enable words with both types of affixes to be processed by this stemmer system. This testing element would be able to handle just one part of the integration, which is the integration of a single prefix with a single suffix while other types of integrations would be handled in the next testing module. This is discussed in the following section.

There are principally two types of test data that are used in this testing phase. First, the test data for words with core prefix and integrated suffix would be tested. The results of this testing are as follows:

Test Data (Prefix + Suffix)	Word	Stem1	Stem2	Ind
Words starting with '*ber*' and ending with a suffix	**ber**kejar**an**	kejar	kejar	✓
Words starting with '*per*' and ending with a suffix	**per**kata**an**	kata	kata	✓
Words starting with '*ter*' and ending with a suffix	**ter**ikut**kan**	ikut	ikut	✓
Words starting with '*mem*' and ending with a suffix	**mem**beza**kan**	beza	beza	✓
Words starting with '*pem*' and ending with a suffix	**pem**batal**an**	batal	batal	✓
Words starting with '*menge*' and ending with a suffix	**menge**cat**kan**	cat	cat	✓
Words starting with '*penge*' and ending with a suffix	**penge**boma**n**	bom	bom	✓
Words starting with '*meng*' and ending with a suffix	**meng**hindar**i**	hindar	hindar	✓
Words starting with '*peng*' and ending with a suffix	**peng**hasil**an**	hasil	hasil	✓
Words starting with '*men*' and ending with a suffix	**men**jatuh**i**	jatuh	menjatuhi	×
Words starting with '*pen*' and ending with a suffix	**pen**cemar**an**	cemar	cemar	✓
Words starting with '*me*' and ending with a suffix	**me**layar**i**	layar	layar	✓
Words starting with '*pe*' and ending with a suffix	**pe**maju**an**	maju	aju	×
Words starting with '*be*' and ending with a suffix	**be**raja**kan**	raja	raja	✓
Words starting with '*ke*' and ending with a suffix	**ke**bolehan	boleh	boleh	✓
Words starting with '*se*' and ending with a suffix	**se**bilang**an**	bilang	bilang	✓
Words starting with '*te*' and ending with a suffix	**te**rasa**kan**	rasa	rasa	✓
Words starting with '*di*' and ending with a suffix	**di**asas**kan**	asas	asas	✓

Table 6.5 Test Data and Result for Prefix-Suffix Stemming

The first-prefix-then-suffix stemming function works out pretty much well (as can be seen from the above table) except for two isolated cases:

1. The first defect is the inability of this stemmer to stem the word '*menjatuhi*'. The cause for this flaw was pretty much hard to trace down as the stemming function for both the prefix '*men*' and the suffix '*i*' works fine; without any shortcomings in the previous testing phase. After a thorough analysis process, the fault was found to be in the database (word dictionary). The root word '*jatuh*' was somehow missing from the database.

   Correction

   As this was a database creation error, it was a bit easier to solve (although it was harder to figure out) if compared to solving a coding problem. The omitted root word '*jatuh*' was eventually entered into the database and the whole problem has been solved. When this system is re-tested after then, it produces the correct root word of '*jatuh*'.

2. The second defect arises from the incomplete correction made to the coding after the unit testing process. The problem lies with the confusion yet again. The system could not differentiate between words starting with the prefix '*pe*' (the root word begins with the letter '*m*') and words starting with the prefix '*pem*'. Thus, the prefix '*pem*' (and not the prefix '*pe*') was removed from the word '*pemajuan*' which results in the incorrect output of '*aju*'.

Correction

The checking module is included in the stemming function of the prefix '*pem*' to solve this problem. Words beginning with the string '*pem*' are checked as to verify whether the prefix is '*pem*' or '*pe*'. When this has been resolved, the stemming process continues as in an usual situation.

The first type of test data have been analyzed above and now, the testing process continues with the second test data, which is the test data for words with core suffix and integrated prefix. The results of this testing are as follows:

Test Data (Suffix + Prefix)	Word	Stem1	Stem2	Ind
Words starting with a prefix and ending with '*nya*'	**ke**semua**nya**	semua	semua	✓
Words starting with a prefix and ending with '*kan*'	**men**dapat**kan**	dapat	dapat	✓
Words starting with a prefix and ending with '*an*'	**per**lakua**n**	laku	laku	✓
Words starting with a prefix and ending with '*i*'	**di**patuh**i**	patuh	patuh	✓
Words starting with a prefix and ending with '*kah*'	-	-	-	-
Words starting with a prefix and ending with '*lah*'	-	-	-	-
Words starting with a prefix and ending with '*pun*'	-	-	-	-
Words starting with a prefix and ending with '*ita*'	-	-	-	-
Words starting with a prefix and ending with '*man*'	-	-	-	-
Words starting with a prefix and ending with '*wan*'	-	-	-	-
Words starting with a prefix and ending with '*wati*'	-	-	-	-

Table 6.6 Test Data and Result for Suffix-Prefix Stemming

The above table signifies that the first-suffix-then-prefix stemming function works fine without facing any significant deficiencies.

Integration of More Than Two Affixes

As mentioned in the previous section, this testing phase will concentrate on the integration types other than that of the first case. Words with two prefixes or two suffixes would be handled in this testing module. The test data developed for this testing module also will be able to handle words with three affixes (words with two prefixes and a suffix as well as words with two suffixes and a prefix). The results of this testing are as follows:

Test Data (>2 Affixes)	Word	Stem1	Stem2	Ind
Words with two prefixes	**diper**buat	buat	buat	✓
Words with two suffixes	bersih**kannya**	bersih	bersih	✓
Words with two prefixes and one suffix	**memper**baiki	baik	baik	✓
Words with two prefixes and one suffix	**berke**bolehan	boleh	boleh	✓
Words with one prefix and two suffixes	**men**jalan**inya**	jalan	jalan	✓
Words with one prefix and two suffixes	**di**sedia**kannya**	sedia	sedia	✓

Table 6.7 Test Data and Result for Combination Stemming

The above table indicates that the integrated stemming function which handles words with two or more affixes is working properly.

6.5.3 System Testing

The testing process that has been performed up until now involves different parts and components of this stemmer system. Although all of these components have been tested, the testing has been carried out separately and independently for each of the components. Moreover, all the components are only tested with words, and not sentences. As such, this testing phase will attempt to test the system as a whole. All the components will be linked up to structure the stemmer system, which will then be used to take in a sentence and stems that sentence, by removing any attached affixes. The results of this testing are as shown in Table 6.8.

6.5.4 User Testing

System testing has been executed and now it is the turn of user testing. As suggested by its name, user testing will be carried out by users of that particular system, for whom the system was developed for. The system developer uses his own test data to test the system in the system testing process but in this user testing procedure, the test data will be provided by the users. This contributes to the difference between system testing and user testing. As for this stemmer system, unfortunately, this type of testing has not been performed due to lack of time. However, the user testing process could be carried out in later stages or in the near future as to verify and validate this system's functionality from a user's perspective.

Test Data (Sentence)	Test Results	Ind
ahmad berjalan kaki ke perpustakaan	Target Output ahmad jalan kaki ke pustaka System Output ahmad jalan kaki ke pustaka	✓
pelajar-pelajar universiti akan menduduki peperiksaan pada minggu ini	Target Output ajar universiti akan duduk periksa pada minggu ini System Output ajar universiti akan duduk periksa pada minggu ini	✓
separuh daripada hartanya didermakan kepada rumah anak-anak yatim	Target Output paruh daripada harta derma kepada rumah anak yatim System Output paruh daripada harta derma kepada rumah anak yatim	✓
sofia memerlukan dua buah beg untuk dibawa ke perkhemahan	Target Output sofia perlu dua buah beg untuk bawa ke khemah System Output sofia perlu dua buah beg untuk bawa ke khemah	✓
mengenali sesama sendiri adalah amat perlu untuk mewujudkan persefahaman mutlak	Target Output kenal sama sendiri adalah amat perlu untuk wujud faham mutlak System Output kenal sama sendiri adalah amat perlu untuk wujud faham mutlak	✓

Table 6.8 Test Data and Result for Sentence Stemming

The above table indicates that this stemmer system works well as a whole, by stemming a given sentence without any desirable fault.

6.6 Summary

In this chapter, the process of system testing has been introduced and its purpose has been explained in great detail. System testing is actually a process of verification and validation of a developed system. Therefore, it serves as a critical element of software quality assurance and represents the ultimate review of system implementation. Next, the testing strategy which has been used in this testing process was explained. This is then followed by description about the testing manual.

Discussion was also centered at the types of testing which have been performed to this stemmer system. The next section focused on the proper testing process and its subsequent results. There were a few defects that were discovered during this testing process and all these defects were fixed accordingly. The resultant system is hoped to be able to meet all of its requirements and to exercise its full functionality, without any notable shortcomings.

CHAPTER SEVEN

SYSTEM EVALUATION

7.1 Introduction

System evaluation can be well-defined as methods and techniques to evaluate the developed system whether it achieves all the desired functional and non-functional requirements. Besides that, system evaluation is done to evaluate the effectiveness and efficiency of the developed system as well as to ensure that it achieves all the goals and objectives of the system.

System evaluation can also be characterized as a way of extensive testing process. It is sometimes referred to as an extension or expansion of the system testing procedure itself. As such, some of the system evaluation aspects that are to be described in this chapter do involve testing processes as well. Typically, there are two types of system evaluation that could be performed on an implemented system; namely system evaluation proper and also interface evaluation. Both of these evaluation types are handled and described in this chapter.

Due to the testing nature of the evaluation process, the final products or results of this evaluation process would most probably reveal further errors and defects that occur in this system. These undetected defects from the previous system testing process would be discovered in this phase and this highlights the importance of system evaluation. These flaws would then be corrected or rectified in order to improve the overall

functionality of this system. Alternatively, changes could be made to this stemmer project in later stages and these changes are proposed as future enhancements to this stemmer, as explained in the next chapter.

Another advantage of system evaluation is that it offers opportunity to assess the system strengths and advantages. Several new strengths could be discovered by carrying out evaluation process. On the other hand, new constraints and limitations could be revealed as well through this system evaluation.

7.2 System Evaluation

As mentioned earlier, system evaluation is an extension of system testing. As such, this two processes may overlap but with a difference – system evaluation is carried out in a more extensive and thorough manner. As for this *Bahasa Melayu Stemmer*, the system evaluation process has been performed by testing this stemmer with a number of sentences which were obtained from *Berita Harian* online newspaper. This is done as to evaluate the performance of this stemmer against real-life data, or in this case, real-world applicable sentences.

Sentences from different fields of interest have been used for the evaluation purpose and this is again due to reflect the diversity of the evaluation task undertaken. The evaluation process and its outcome are presented below:

Evaluation 1

As a way to start off this evaluation process, a paragraph of sentences, which are obtained from *Berita Harian* online newspaper website, are tested with this stemmer. The text used for this part of evaluation is categorized as entertainment news and the sentences are as follows:

> Penyanyi jelita Beyonce Knowles menyapu bersih lima Anugerah Grammy ke-46 di Staples Center, Los Angeles pagi Ahad lalu untuk album solo pertamanya Dangerously In Love manakala Justine Timberlake membolot dua anugerah.

The evaluation result for the above text is as follows:

> nyanyi jelita beyonce knowles sapu bersih lima anugerah grammy ke di staples center, los angeles pagi ahad lalu untuk album solo pertama dangerously in love manakala justine timberlake bolot dua anugerah.

From the above evaluation result, it can be summarized as:

- Total number of words : 31
- Correctly stemmed words : 30
- Evaluation percentage : 97%

Evaluation 2

The text used for this part of evaluation is categorized as national news and the sentences are as follows:

> Berinspirasikan keupayaannya mengeluarkan model sendiri menerusi Waja dan iltizam jitunya untuk terus berubah, Perusahaan Otomobil Nasional Bhd (Proton) memperkenalkan generasi kedua kereta yang dibangunkannya sendiri menerusi jentera yang dikenali sebagai Gen.2.

The evaluation result for the above text is as follows:

> inspirasi upaya keluar model sendiri terus waja dan iltizam jitu untuk terus berubah, usaha otomobil nasional bhd kenal generasi dua kereta yang bangun sendiri terus jentera yang kenal bagai gen.2.

From the above evaluation result, it can be summarized as:

- Total number of words : 31
- Correctly stemmed words : 29
- Evaluation percentage : 94%

Evaluation 3

The text used for this part of evaluation is categorized as national news and the sentences are as follows:

> Datuk Seri Abdullah Ahmad Badawi melancarkan kempen antimerokok besar-besaran di sini semalam dengan meminta penguatkuasaan undang-undang yang melarang penjualan rokok kepada remaja berusia bawah 18 tahun dipertingkatkan.

The evaluation result for the above text is as follows:

> datuk seri abdullah ahmad badawi lancar kempen anti rokok besar di sini malam dengan pinta kuatkuasa undang yang larang jual rokok kepada remaja usia bawah 18 tahun tingkat.

From the above evaluation result, it can be summarized as:

- Total number of words : 27
- Correctly stemmed words : 27
- Evaluation percentage : 100%

Evaluation 4

The text used for this part of evaluation is categorized as international news and the sentences are as follows:

> Seorang wanita 23 tahun didapati masih hidup selepas dikeluarkan daripada runtuhan pangsapuri 11 tingkat di tengah Turki semalam, tujuh hari selepas bangunan itu runtuh, lapor televisyen swasta NTV. Pekerja penyelamat berusaha menyelamatkan Yasemin Yaprakci daripada runtuhan awal pagi semalam dan mendapati wanita itu dalam keadaan baik kecuali kakinya terperangkap dalam runtuhan batu.

The evaluation result for the above text is as follows:

> orang wanita 23 tahun dapat masih hidup lepas keluar daripada runtuh pangsapuri 11 tingkat di tengah turki semalam, tujuh hari lepas bangun itu runtuh, lapor televisyen swasta ntv. kerja selamat usaha selamat yasemin yaprakci daripada runtuh awal pagi malam dan dapat wanita itu dalam ada baik kecuali kaki perangkap dalam runtuh batu.

From the above evaluation result, it can be summarized as:

- Total number of words : 52
- Correctly stemmed words : 51
- Evaluation percentage : 98%

Evaluation 5

The text used for this part of evaluation is categorized as international news and the sentences are as follows:

> Seorang remaja lelaki 17 tahun Britain bertindak mencuri kad kredit bapanya dan membelanjakan lebih 2,000 (RM72,000) dengan bersuka ria di Rome, Itali selama empat hari. Akhbar The Sun melaporkan semalam, remaja itu, Tom Smith mencuri kad itu ketika bapanya keluar berjoging. Dia kemudiannya, menempah tiket kapal terbang dari Stansted ke Rome dan bermalam sehari di sebuah hotel tiga bintang di ibu negara Itali itu.

The evaluation result for the above text is as follows:

> orang remaja lelaki 17 tahun britain tindak curi kad kredit bapa dan belanja lebih 2,000 (rm72,000) dengan suka ria di rome, itali lama empat hari. akhbar the sun lapor semalam, remaja itu, tom smith curi kad itu ketika bapa keluar berjoging. dia kemudiannya, tempah tiket kapal terbang dari stansted ke rome dan malam hari di buah hotel tiga bintang di ibu negara itali itu.

From the above evaluation result, it can be summarized as:

- Total number of words : 66
- Correctly stemmed words : 63
- Evaluation percentage : 95%

Evaluation 6

The text used for this part of evaluation is categorized as economics news and the sentences are as follows:

> Majlis Tindakan Ekonomi Negara (MTEN) mahu kedua-dua pihak Malaysia dan Singapura menghentikan pertikaian berhubung isu air dan menyelesaikannya secara harmoni dan saksama. MTEN dalam iklan terakhirnya yang disiarkan di akhbar tempatan hari ini menjelaskan, apa yang Malaysia mahu hanyalah harga yang berpatutan serta satu formula penyelesaian yang saksama dan sejati bagi manfaat kedua-dua negara.

The evaluation result for the above text is as follows:

> majlis tindak ekonomi negara (mten) mahu dua pihak malaysia dan singapura henti tikai hubung isu air dan selesai cara harmoni dan saksama. mten dalam iklan akhir yang siar di akhbar tempat hari ini menjelaskan, apa yang malaysia mahu hanya harga yang patut serta satu formula selesai yang saksama dan sejati bagi manfaat dua negara.

From the above evaluation result, it can be summarized as:

- Total number of words : 54
- Correctly stemmed words : 52
- Evaluation percentage : 96%

Evaluation 7

The text used for this part of evaluation is categorized as technology news and the sentences are as follows:

> Produk terkini Sony, Cyber-shot F828 menjadi kamera digital pertama dibekalkan dengan resolusi lapan mega piksel berserta ciri tapisan empat warna CCD dan lensa terkini Carl Zeiss Vario Sonnar T sesuai untuk pengguna profesional. Cyber-shot F828 kini boleh didapati di semua pengedar bertauliah Sony dengan harga RM4,199.

The evaluation result for the above text is as follows:

> produk kini sony, cyber-shot f828 jadi kamera digital pertama bekal dengan resolusi lapan mega piksel serta ciri tapis empat warna ccd dan lensa kini carl zeiss vario sonnar t sesuai untuk guna profesional. cyber-shot f828 kini boleh dapat di semua edar tauliah sony dengan harga rm4,199.

From the above evaluation result, it can be summarized as:

- Total number of words : 46
- Correctly stemmed words : 46
- Evaluation percentage : 100%

Evaluation 8

The text used for this part of evaluation is categorized as computer news and the sentences are as follows:

> Virus komputer yang sedang melanda Internet masa kini, MyDoom, dijangka akan terus menyerang e-mel di seluruh dunia sehingga 12 Februari ini, tarikh ia diprogramkan untuk berhenti mengganas. Virus itu yang juga dikenali dengan Novarg, kini sudah menyerang berjuta komputer pengguna di seluruh dunia tetapi angka sebenarnya tidak dapat dipastikan.

The evaluation result for the above text is as follows:

> virus komputer yang sedang landa internet masa kini, mydoom, jangka akan terus serang di seluruh dunia hingga 12 februari ini, tarikh ia program untuk henti mengganas. virus itu yang juga kenal dengan novarg, kini sudah serang juta komputer guna di seluruh dunia tetapi angka benar tidak dapat dipastikan.

From the above evaluation result, it can be summarized as:

- Total number of words : 49
- Correctly stemmed words : 47
- Evaluation percentage : 96%

Evaluation 9

The text used for this part of evaluation is categorized as sports news and the sentences are as follows:

> Harga tiket antara RM7 hingga RM30 dikenakan kepada peminat untuk menyaksikan pertandingan kelayakan badminton Piala Thomas/Uber 2004 yang akan berlangsung dari 16 Februari hingga 22 Februari ini di Stadium Badminton Kuala Lumpur (KLBA), Cheras.

The evaluation result for the above text is as follows:

> harga tiket antara rm7 hingga rm30 kena kepada minat untuk saksi tanding layak badminton piala thomas/uber 2004 yang akan langsung dari 16 februari hingga 22 februari ini di stadium badminton kuala lumpur (klba), cheras.

From the above evaluation result, it can be summarized as:

- Total number of words : 34
- Correctly stemmed words : 34
- Evaluation percentage : 100%

7.3 Summary

System evaluation is a continuation process from the previous phase, system testing. System evaluation is done to evaluate the performance of the developed system as well as to ensure its effectiveness and efficiency. Overall functionality of the system can be assessed as well by implementing evaluation process.

As system evaluation ensures completeness of a system, this has been performed on this *Bahasa Melayu Stemmer* as well. Sample sentences have been obtained from the *Berita Harian* online newspaper website and these sentences are tested using this stemmer. Variety of sentences which reflects different types of news have been used for the evaluation process; ranging from international news to entertainment news and even including sports or economics news.

All these sentences yield an evaluation percentage of over 90% - this indicates the reasonably excellent performance of this stemmer. The diversity in the evaluation process undertaken coupled with its overwhelming result is hoped to further ensure the completeness and comprehensiveness of this stemmer.

CHAPTER EIGHT

DISCUSSION AND CONCLUSION

8.1 Introduction

The proper development of this stemmer system has been discussed in depth in the previous chapters, starting from system requirements until system evaluation. Each phases of the system development has been explained in great detail. Thus, this very last chapter will attempt to provide some further information regarding this developed system. A few final and concluding remarks will be presented here as to serve as a fundamental approach in building a stemmer.

Problems encountered throughout this development process would be discussed here and a few solutions or steps taken to overcome these problems are recommended as well. Then, the discussion would be shifted to focus on the system strengths where the advantages of this system would be discussed. This is then followed by presenting some of the system limitations which were identified.

A few future enhancements were also proposed for this stemmer project as to enhance its overall functionality. These enhancements would be vital in upholding the future advancements of this developed system. Lastly, knowledge and experiences gained throughout this development process are discussed as well. This discussion is hoped to provide one with an overview of the entire development process which have been carried out successfully all the way through.

8.2 Problems Encountered and Solutions

All projects and systems do have their own problems and difficulties regardless of the development approaches taken. This *Bahasa Melayu Stemmer* is not exceptional after all. Right from the system specification until the implementation process, I have encountered a number of problems. While a few of these problems are easily solved, others proved to be substantially tougher to be unraveled. The problems encountered throughout and its solutions are presented below:

Difficulties in determining the appropriate development tools

- This problem arises at the beginning stage itself when I had to choose a development tool to implement my system. There were a lot of development tools available in the market, especially the fourth-generation programming languages, such as Visual Basic, Java, C++ and Prolog. All the available tools have their own strengths and weaknesses so choosing the most suitable tool is critical in ensuring the accomplishment of this system.

- Solution

  I obtained some valuable views from my supervisor and lecturer regarding the choice of development tools. Apart from that, I also analyzed other stemmers to get an overview of its development tools. From this gathered information, I opted for C++ as my programming tool and it proves to be a good choice after all, looking at this successfully implemented system.

Difficulties in determining the scope of the system

- The system requirements for this stemmer is pretty much straightforward – as it is a stemmer, it should be able to stem words and sentences. However, I found it difficult to come up with a well-defined scope for this stemmer. Due to the interdependence nature of stemmers in general, outlining the scope for this system proved to be difficult. The inability to distinguish which of the system features are essential, or optional, has made defining the project scope a complicated process. I was not sure whether the stemming should be limited to words or should it be able to stem sentences as well. Another tricky question was whether to handle the so-called 'kata ganda' and infixes in Malay language. All these arisen difficulties can be attributed to my inexperience in developing similar systems beforehand.

- Solution

  Again, my supervisor and moderator were there to provide me with precious and priceless advice regarding this matter. I also went through a few related papers (on the stemming topic particularly) to get a general idea of stemming. This helped me in determining the project scope. I choose to handle a sentence as my system's scope and I also decided to include 'kata ganda' and infixes in my stemmer. Although this seemed tougher to implement at the beginning, I took the challenge to put it into practice at that time and I really think that I have achieved it now with overwhelming success.

Lack of resources

- Information gathering is one of the most important phases of any project as it is in this stemmer system. However, finding information for this *Bahasa Melayu Stemmer* proved to be an easier said than done process as there were lack of resources in this related area. Information about general stemming, particularly in English language, is widely available. However, there was very few information which can be obtained for stemming process in Malay language. Malay stemmers are rarely to be seen and Malay stemming algorithms are too hard to be found on the Internet.

- Solution

  As information gathering via Internet was not going anywhere, I had tried other alternatives in order to gather the needed information. Other methods of acquisition was used such as reference to research papers and books as well as having informal interviews with experienced experts.

Time constraint

- I found that the time given to me to develop this system was very limited considering the fact that my knowledge in this area of study was rather shallow. I found it quite difficult in the starting stages to juggle my time between classes and developing this system. At the same time, I have to find my time to write this report as well. So much of time were spent on the coding phase, where half of that time was for coding and the rest for figuring out the

errors in the codes. As I was attending other classes, I also have to commit myself to a few other projects and assignments; and this was the reason for the lack of time which I experienced.

❖ Solution

The only solution to this problem was time planning – I planned out my timetable and made it a point to follow my schedule strictly in order to complete my system in time. At first, it was a bit hard but as time passed by, I gradually manage to cope up with this system development and other tasks. I planned all my activities before carrying it out and I also made sure that all these activities are kept according to schedule. This way, it helped me to complete my stemmer system within the allocated time.

8.3 System Strengths

This stemmer system is just one of the few stemmers that are available in the Malay language. The ability to stem Malay texts itself can be considered as an advantage. Apart from that, this stemmer is also capable of stemming 'kata ganda' and also infixes, which is not present in any other languages other than Malay language. This added features can be considered as further strong points for this stemmer system. Besides that, *Bahasa Melayu Stemmer* does possess other additional strengths, and these strengths are summarized in the table below:

System Strength	Explanation
Highly accurate	This stemmer employs dictionary checking in order to stem a given word or text to its correct root form (accurate stemming)
Simple and easy-to-use	The stemming process could be done by just clicking a button (it is as simple as that) and the flow of the system is easy to follow
Enhanced display	The stemmed text with attached affixes are displayed in addition to the stemmed text as this could enhance one's understanding
System transparency	Users can use this system without needing to know about the implementation of the system or how it functions
Maintainability of data	The program is coded in such a way that supports future modification and enhancement in a simplified way (maintainability)

Table 8.1 System Strengths

8.4 System Limitation

Although this *Bahasa Melayu Stemmer* does offer a great number of advantages and strengths, it does come with a cost. There are still a few limitations and constraints to this stemmer system. This occurs merely due to the fact that I have overlooked some of the aspects in the development stages; and this is again due to the lack of time. The following table summarizes the limitations of this system:

System Strength	Explanation
Slower response time	The performance of this stemmer is a major drawback. It is a bit slower compared to other stemmers as it employs file checking (it trades response time with accuracy)
Less attractive use-interface	The user-interface is simple to use but then, it is less attractive and creative. User-interface should have these two characteristics generally in order to attract users

Table 8.2 System Limitation

8.5 Future Enhancements

Due to time and knowledge constraint, there were a few aspects of my system that could not be added in. This accounted for the limitations in my project. I have outlined these identified weaknesses as a future enhancement as to make my stemmer a better one in the future. The future enhancements for *Bahasa Melayu Stemmer* also arise due to the new ideas that emerge during the development process. A few other features could be added in to this stemmer as well as to enhance its functionality. Among the additional features, or future enhancements, that could be implemented in the future are as follows:

- **Improve Response Time**

  As this stemmer employs dictionary checking to ensure the correct root word is returned, it is a bit slower in the term of response time. However, this drawback could be improved in the future by utilizing the indexing concept.

Another solution to this problem would be to have multiple files to perform the word checking as opposed to a single file currently.

- **Improve User-Interface**

  As to attract the users to use this system, the user-interface should be given importance. It should be more attractive and creative in the first place. This could be done by introducing more attractive colours to the interface. Having some minor animation is another option as well.

- **Database Backup and Recovery**

  The database, or word dictionary, is vital for the continuing performance of this stemmer. If it is corrupted, then the whole process of stemming could not take place. As to avoid this unpleasant instance, the database could be backed up by having another similar database; which could be activated in the case if the present one fails as this would ensure data integrity.

- **Online Dictionary**

  This stemmer has the potential to function as an online Malay dictionary as well. The only adaptation to be done is to link this stemmer to a web server and when this has been done, this stemmer is ready to act as an online dictionary for the Malay language.

- **Language Translator**

  This feature is an expansion of the previous one. Once linked to a web server, this system can be integrated with online dictionaries from other languages in

order to operate as a language translator. Alternatively, other languages could be loaded into this system to perform the same function.

- **Text-to-Speech System**

  Stemmers are widely used in the field of information retrieval but speech synthesis (text-to-speech) provides a new area of interest. This stemmer could be integrated with a few other applications to construct a text-to-speech synthesizer. The role of stemmer is principally to solve the 'e pronunciation problem' in this text-to-speech synthesizer.

8.6 Knowledge and Experience Gained

Throughout developing the *Bahasa Melayu Stemmer*, I have gained a great amount of knowledge and experience which will be very beneficial in the future, especially after my studies and when I start stepping into the working world. At a glance, this system development seems just like another project which I had to undertake as part of my studies requirements. However, when I had completed developing this stemmer and when I looked back all the way through, I realized how much of knowledge and experience I have gained so far. There were an enormous amount of knowledge and experience gained throughout but I would just prefer to list down the most prominent ones and this includes:

- Acquired a wider knowledge in the field of stemming. I have improved my knowledge in a number of aspects such as stemming processes, stemming

algorithms and other related topics. I also got an opportunity to research about other stemmers which have been developed previously.

- Attained a great amount of knowledge in the field of Natural Language Processing (NLP). Throughout this system development process especially during system requirements, I have mastered different parts of this subject such as syntax, semantic, morphology and phonology.

- Learned in depth on certain development tools and technology. I managed to improve my C++ programming skills and this includes utilizing built-in functions which I never experimented before. I also got the opportunity to learn about the Microsoft Foundation Class (MFC) application, which is basically used to create user-interfaces.

- Exposed to the literature side of the Malay language. As this system deals with Malay words and sentences, it turned out to be a necessity for me to improve my knowledge on the Malay language, especially on the literature side. I discovered a few new things in the process which I never come across before, even in my schooling days.

- Exposed to the proper way of developing a system, which has to be carried out one process at a time, starting from the system definition until the system evaluation process. Another thing which I realized was the importance of the System Development Life Cycle (SDLC) which outlines the fundamental phases and activities involved in developing a system.

- Learnt how to work under pressure and within a tight time constraint. I had to juggle my time between a lot of tasks, including this system development. It might have been a terrible experience at present but this experience would be valuable for me in the future, preparing me for the worst, when I have to cope up with greater working stress and pressure.

- Utilized my theoretical knowledge that I have acquired during classes or lectures. Through this system development, I was able to put into practice all the theories that I have learned for the past three years. I realized that only through practical works, the importance of all the learnt theories can be seen clearly and understood properly.

- Other than that, I also learned something new, something which has not been taught in classrooms or lecture halls. For instance, the skills of sharing and exchanging knowledge is a wonderful experience which I have been exposed to. I truly believe that communicating knowledge in such a way is something beneficial and valuable for the future.

8.7 Conclusion

This chapter had discussed every inch about this *Bahasa Melayu Stemmer*. Although this chapter is not dedicated to a system development phase as other previous chapters, it does provide some useful explanation and guidelines about this stemmer. A few concluding remarks have been presented here as to enhance one's perceptive and understanding on the proper development of a stemmer.

Starting from system specification right through system evaluation, there were few problems encountered and all these problems have been discussed in great detail in this chapter. The recommended solutions and steps taken to overcome these problems are described as well. Then, the discussion was shifted to look at the system strengths and limitations. In these two sections, the strengths possessed by this stemmer system are highlighted and its limitations are revealed as well.

Besides, this chapter also summarizes the future enhancements that could be coupled up with this stemmer project. The successful development of this stemmer at present is the first step towards the future expansion of the system. The knowledge and experience gained during this system development also would be valuable and beneficial in the future undertakings.

REFERENCE

Books

1. Allec, J. *Natural Language Understanding*, 2nd edition, The Benjamin/Cummings Publishing Company, New York, 1995.

2. Baeza Yates, R. *Word Stemmer Application*, Englewood Cliffs, New Jersey, 1992.

3. George F. Luger & William A. Stubblefield. *Artificial Intelligence*, 3rd edition, Prentice Hall, London, 1998.

4. Jeffrey L. Whitten, Lonnie D. Bentley & Kevin C. Dittman. *System Analysis and Design Methods*, 5th edition, McGraw-Hill, Singapore, 2002.

5. Sommerville, I. *Software Engineering*, 6th edition, Addison Wesley, Harlow, 2001.

Articles

1. Dawson, J.L. *Suffix removal for word conflation*, Bulletin of the Association for Literary & Linguistic Computing, 1974, 2(3), 33-46.

2. Frakes, W.B. *Stemming for Information Retrieval*, PhD. Dissertation, Syracuse University, August 1982.

3. Harman, D. *How Effective is Suffixing?*, Journal of the American Society for Information Science, 1991, 42(1), 7-15.

4. Hull, D.A. *Stemming Algorithms: A Case Study for Detailed Evaluation*, Journal of the American Society for Information Science, 1996, 47(1), 70-84.

5. Lennon, M., Pierce, D.S., Tarry, B.D. and Willett, P. *An Evaluation of Some Conflation Algorithms for Information Retrieval*, Journal of Information Science, 1981, 3, 177-183.

6. Lovins, J.B. *Development of a Stemming Algorithm*, Mechanical Translation and Computational Linguistics, 1968, 11, 22-31.

7. Porter, M.F. *An Algorithm for Suffix Stripping*, Program, 1980, 14, 130-137.

8. Ulmschneider, J. and Doszkocs, T. *A Practical Stemming Algorithm for Online Search Assistance*, Online Review, 1983, 7(4), 42-55.

Websites

1. Stemming
http://www.webmasterworld.com/glossary/stemming.htm

2. Stemmer from FOLDOC
http://www.wombat.doc.ic.ac.uk/foldoc/foldoc.cgi?stemming

3. What is Stemming?
http://www.comp.lancs.ac.uk/computing/research/stemming/general/

4. Porter Stemming Algorithm
http://www.tartarus.org/~martin/PorterStemmer/

5. Porter Stemming Algorithm
http://sbs.cs.olemiss.edu/345/stem10.html

6. Lovins Stemmer
http://www.hkn.eecs.berkeley.edu/~dyoo/python/py_lovins/

7. The Lancaster (Paice/Husk) Stemming Algorithm
http://www.comp/lancs.ac.uk/computing/research/stemming/

8. Stemming Algorithms

http://www.mis.nsysu.edu.tw/~syhwang/Courses/StemmingAlgorithms.htm

9. Stemming Algorithms

www.tartarus.org/~richard/xapian/docs/stemming.html

10. Stemming Performance

http://www.comp.lancs.ac.uk/computing/research/stemming/perfomance.htm

11. Application of Stemmer

http://www.mds.rmit.edu.au/mg/mods/mod10.html

12. Future of Stemming

http://orange.cs.umass.edu/whatsnew/stemming.html

APPENDIX

APPENDIX A – SAMPLE CODES

Class Declaration

```
class Stemmer
{
private:
      string sentence;
      string word[NO_OF_WORDS];
      int flag[NO_OF_WORDS];
public:
      Stemmer();
      void setSentence(int);
      void setWord(string, int);
      void setFlag(int);
      void stem(string, int);
      void stemExtra(int);
      void stemDouble(int);
      void stemInfix(int);
      void stemException(int);
      void stem_it_out(string);
      bool check(string);
      bool isVowel(char);
      int getFlag(int);
      string getWord(int);
      string getSentence();
};
```

Class Definition

```
Stemmer::Stemmer()
{
      for(int i=0 ; i<NO_OF_WORDS ; i++)
      {
            flag[i] = 0;
      }
}

void Stemmer::setSentence(int w)
{
      for(int i=0 ; i<=w ; i++)
      {
            sentence += word[i];
            sentence += " ";
      }

      sentence.erase(sentence.length());
}

void Stemmer::setWord(string s, int i)
{
      word[i].assign(s);
}

void Stemmer::setFlag(int i)
{
      flag[i] = 1;
}

void Stemmer::stem(string s, int i)
{
      if(check(s))
      {
            setWord(s,i);
            setFlag(i);
      }

      else
      {
            int d = s.length() - 1;

            //remove prefix

            //remove 'ber'
            if((s[0] == 'b') && (s[1] == 'e') && (s[2] == 'r'))
            {
                  string temp(s);
```

```
        //remove be***
        if(check(temp.substr(2,string::npos)))
        {
              s.erase(0,2);
              setWord(s,i);
              setPrefix(temp.substr(0,2),i);
              setFlag(i);
        }

        //remove ber***
        else
        {
              s.erase(0,3);
              setPrefix(temp.substr(0,3),i);
              stem(s,i);
        }
}

//remove 'mem'
else if((s[0] == 'm') && (s[1] == 'e') &&
       (s[2] == 'm'))
{
      string temp(s);

      if(!isVowel(s[3]))
      {
            s.erase(0,3);
            setPrefix(temp.substr(0,3),i);
            stem(s,i);
      }

      else
      {
            s.erase(0,2);
            setPrefix(temp.substr(0,2),i);
            stem(s,i);
      }
}

//remove 'meny'
else if((s[0] == 'm') && (s[1] == 'e') &&
       (s[2] == 'n') && (s[3] == 'y'))
{
      string temp(s);

      //remove me***
      if(check(temp.substr(2,string::npos)))
      {
            s.erase(0,2);
            setWord(s,i);
```

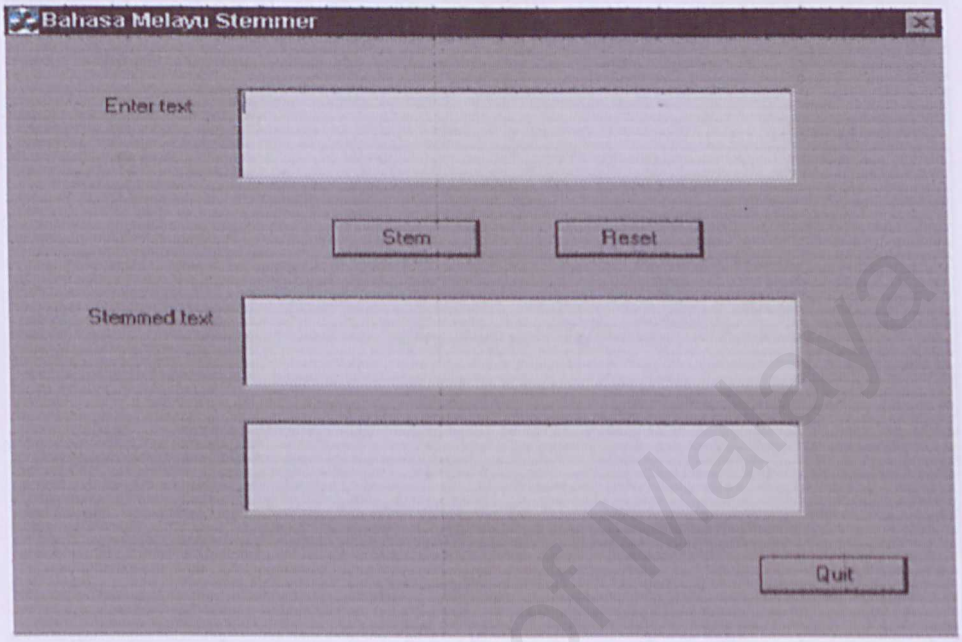
```
            setPrefix(temp.substr(0,2),i);
            setFlag(i);
        }

        //remove meny***
        else
        {
            s.erase(0,3);
            setPrefix(temp.substr(0,3),i);
            stem(s,i);
        }
}

//remove 'men'
else if((s[0] == 'm') && (s[1] == 'e') &&
        (s[2] == 'n'))
{
        string temp(s);

        if(!isVowel(s[3]))
        {
            s.erase(0,3);
            setPrefix(temp.substr(0,3),i);
            stem(s,i);
        }

        else
        {
            s.erase(0,2);
            setPrefix(temp.substr(0,2),i);
            stem(s,i);
        }
}

//remove suffix

//remove 'nya'
else if((s[d-2] == 'n') && (s[d-1] == 'y') &&
        (s[d] == 'a'))
{
        s.erase(d-2);
        setSuffix("nya",i);
        stem(s,i);
}

//remove 'kan'
else if((s[d-2] == 'k') && (s[d-1] == 'a') &&
        (s[d] == 'n'))
{
        s.erase(d-2);
```

```
                setSuffix("kan",i);
                stem(s,i);
        }

        //remove two-letter prefix

        //remove 'di'
        else if((s[0] == 'd') && (s[1] == 'i'))
        {
                s.erase(0,2);
                setPrefix("di",i);
                stem(s,i);
        }

        //remove 'ke'
        else if((s[0] == 'k') && (s[1] == 'e'))
        {
                s.erase(0,2);
                setPrefix("ke",i);
                stem(s,i);
        }

        //remove suffix i
        else if(s[d] == 'i')
        {
                string temp(s);

                if(check(temp.substr(0,d)))
                {
                        s.erase(d);
                        setWord(s,i);
                        setSuffix("i",i);
                        setFlag(i);
                }
        }

        //remove suffix 'an'
        else if((s[d-1] == 'a') && (s[d] == 'n'))
        {
                string temp(s);

                if(check(temp.substr(0,d-1)))
                {
                        s.erase(d-1);
                        setWord(s,i);
                        setSuffix("an",i);
                        setFlag(i);
                }
        }
    }
}
```

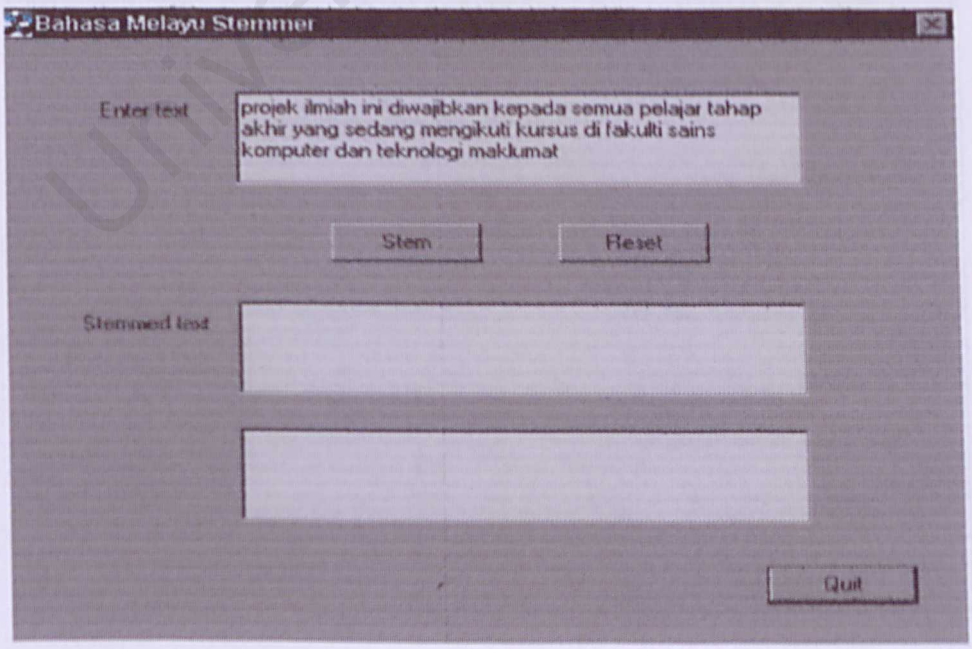
```
bool Stemmer::check(string s)
{
     char str1[30];

     ifstream file("bma-z.txt", ios::in);

     while(file && !file.eof())
     {
          file.getline(str1, 30);

          string temp(str1);

          if(s.compare(temp) == 0)
               return true;
     }

     return false;
}

bool Stemmer::isVowel(char c)
{
     if ((c == 'a') || (c == 'e') || (c == 'i') ||
        (c == 'o') || (c == 'u'))
          return true;

     return false;
}

int Stemmer::getFlag(int i)
{
     return flag[i];
}

string Stemmer::getWord(int i)
{
     return word[i];
}

string Stemmer::getSentence()
{
     return sentence;
}
```

APPENDIX B – USER MANUAL

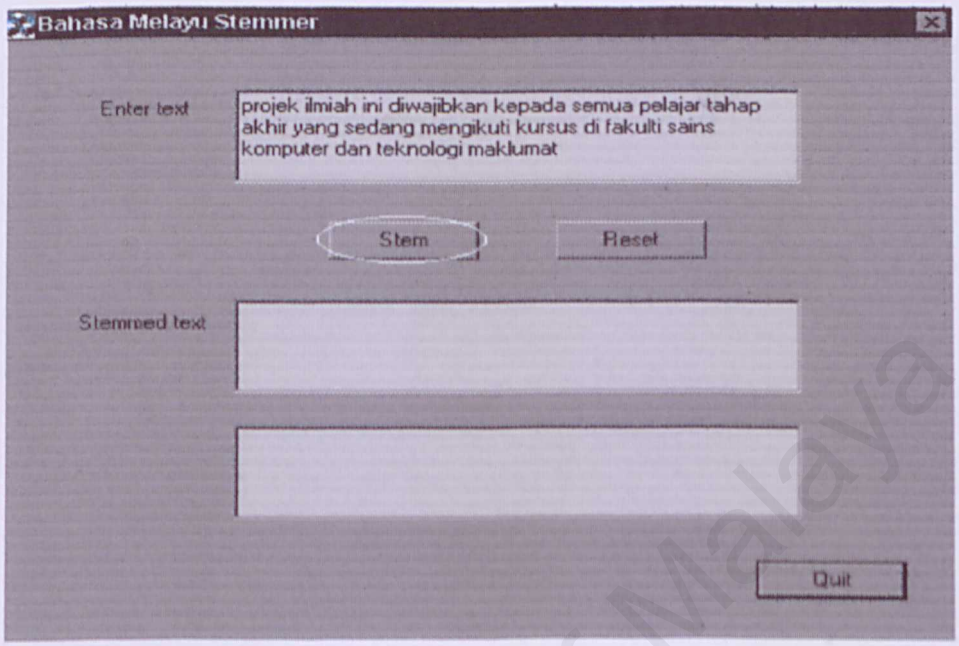
1. Click on the *Bahasa Melayu Stemmer* application icon to open the program. Once clicked, a screen as below will be displayed.



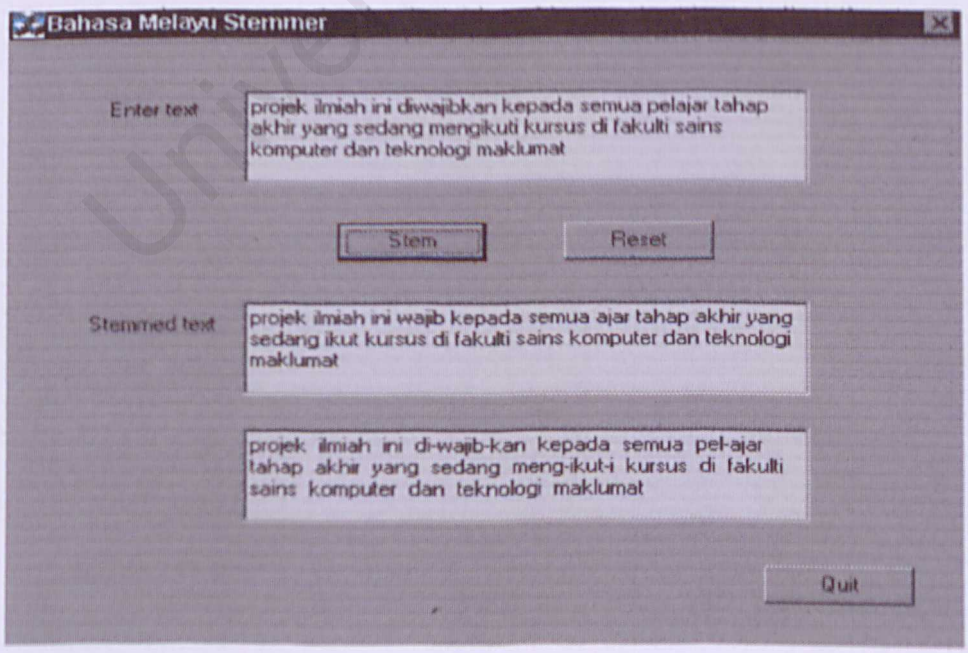
2. The system is ready to stem now. To start off the stemming process, enter a text in the first text box provided at the top of the screen.



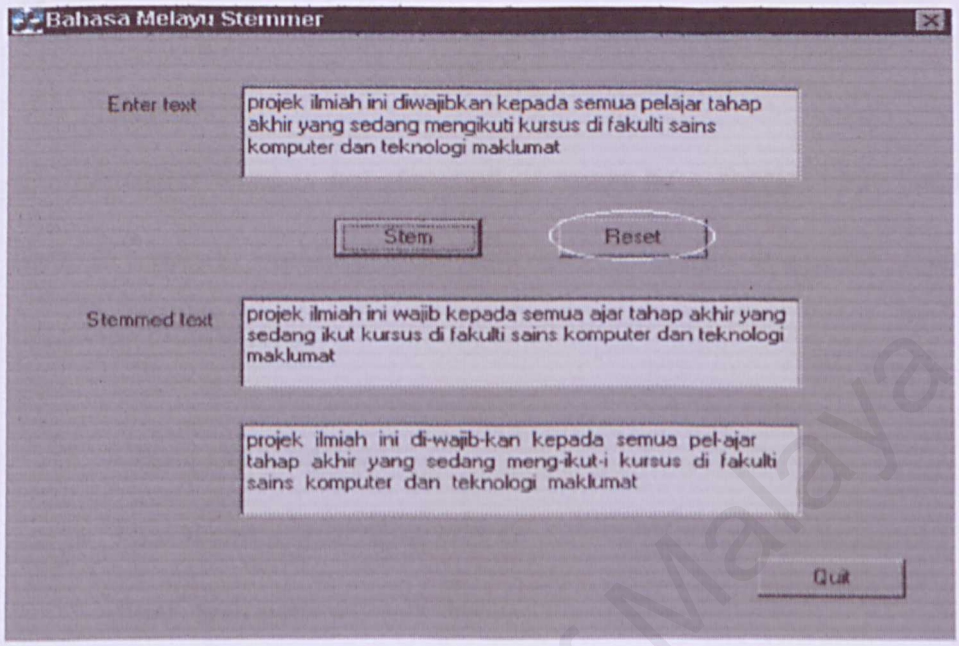
3. When the text has been typed in, click on the '*Stem*' button to stem the entered text. The '*Stem*' button can be found below the first text box and it is on the left-hand side of the screen.



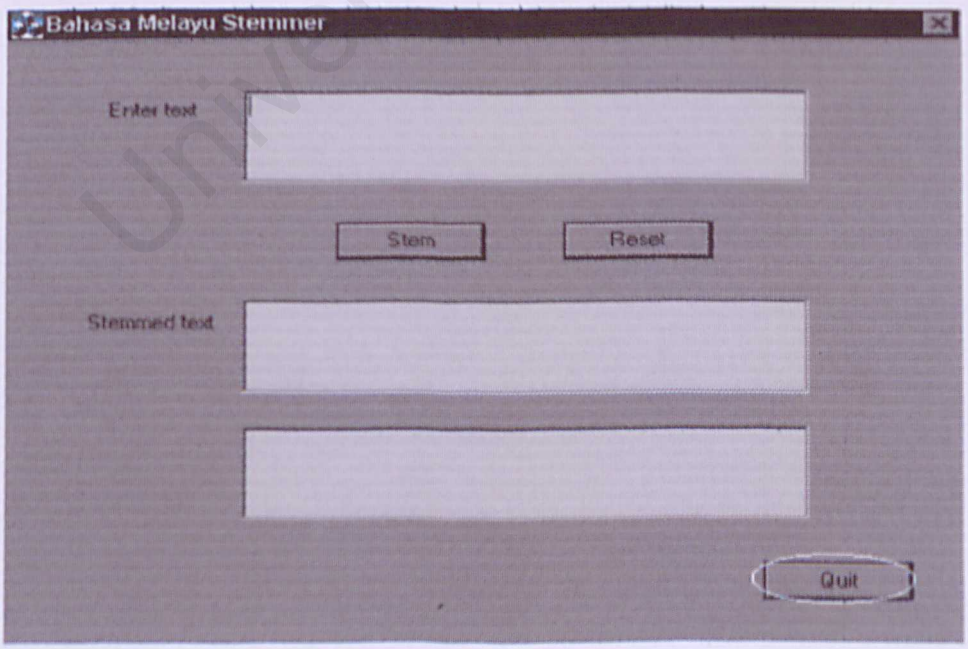
4. This will activate the stemming function to be performed on the entered text. The resultant stemmed text will appear in the two text boxes which are located at the bottom of the screen.



5. Click on the 'Reset' button to clear the contents of all the three text boxes. The 'Reset' button can be found below the first text box and it is situated on the right-hand side of the screen.



6. The resultant screen would look like the one which has been just opened. Repeat the instructions above to continue stemming or click on the 'Quit' button to end the program. The 'Quit' button is at the bottom of the screen.



A Few Important Things To Remember

- This stemmer would just take in words – complete words. So try to avoid typing in numbers or special characters as this would not be recognized by this stemmer. Abbreviations should be avoided as well.

- If a particular word is not stemmed, it might be a special noun such as the name of a person, building, place, city or even country. If it is not, then try the following alternatives:

  - ✓ **Check the language**

    This stemmer just accepts texts in Malay language. Keying in texts in other languages will not work.

  - ✓ **Check the word**

    Check the spelling of the word. The inability of this stemmer to stem a particular word may be due to typo error.