

JTAGGER

By

NORHANA BT YAACOB

Dissertation Submitted in Partial Fulfillment of
the Requirements for the Degree
Bachelor of Technology (Hons)
(Business Information System)

JUNE 2006

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

6

LA

1026.5

.N822

2006

- 1) Computer - assisted instructor
- 2) Education -- Computer programs

CERTIFICATION OF APPROVAL


JTagger

by

Norhana bt Yaacob

A project dissertation submitted to the
Business Information System Programme
Universiti Teknologi PETRONAS
In partial fulfillment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION SYSTEM)

Approved by,



(Ms. Norshuhani Binti Zamin)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
January 2006

CERTIFICATION OF ORIGINALITY

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



NORHANA BT YAACOB

ABSTRACT

Part-of-speech tagging, also called grammatical tagging, is the process of assigning the words in a text with their corresponding parts of speech like noun, verb, pronoun, or other lexical class markers to each word in a sentence. Part-of-speech tagging is an important step in natural language processing.

Part-of-speech tagging is an ambiguous process because a word can represent more than one part of speech at different times. Most difficult task is because it deals with ambiguities of the word. A word, phrase, or sentence is ambiguous if it has more than one meaning. The word 'light', for example, can mean not very heavy or not very dark.

There are two types of ambiguity which are lexical and structural. When a word has more than one meaning, it is said to be lexically ambiguous. When a phrase or sentence can have more than one structure it is said to be structurally ambiguous.

The part-of-speech tagging algorithms fall into three classes which are rule-based taggers, stochastic taggers, and transformation-based taggers. In this project, rule-based tagging algorithm is used as the mechanism to develop the system which named JTagger. The tagger initially tags by assigning each word its most likely tag, estimated by examining a corpus that consists of Penn Treebank Tagsets.

JTagger is automatically performed the tagging process giving reasonable accuracy thus eliminate the difficulties of hand tagging task for the reader to manually tag a sentence. Part-of-speech tagging is important since it could help people to understand English better.

The programming language used in this system is Java because it is an independent source that can run in any platform including Microsoft or UNIX.

ACKNOWLEDGEMENT

My greatest praise to Allah the Almighty that has given me the strength and guide me to finish my Final Year Project.

I would like to express thousands of appreciations, highest gratitude and sincere thanks to my supervisor, Mrs Norshuhani bt Zamin for all the valuable guidance, positive and constructive critics and advice that have been given to me upon completing this project.

I would also like to express my gratitude and thanks to all lecturers and tutors in Information Communication Technology (ICT) and Business Information System (BIS) department who eventually helped me during the project and also in sharing their knowledge and information, which has made this project an unforgettable work. Not to forget, special thanks to all my friends who helped and share their knowledge with me during the development of this project.

Lastly, I acknowledge with greatest appreciation to other personnel not mentioned above whom gave me such great support in completing this project successfully and to UTP for giving me a chance to gain knowledge and experiences during the Final Year Project development. A sincere apology from me for all the problems involuntarily caused. All of your kindness and cooperation are highly appreciated and will be fondly remembered.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	vi
LIST OF TABLES	vii
CHAPTER 1	1
1.1 PART OF SPEECH	1
1.2 WORD CLASSES	2
1.3 PART OF SPEECH TAGGING	7
1.4 TAGSET FOR ENGLISH	8
1.4 BACKGROUND OF STUDY	9
1.5 PROBLEM STATEMENT	11
1.6 OBJECTIVES	12
1.7 SCOPE OF STUDY	12
1.8 SIGNIFICANT OF PROJECT	13
CHAPTER 2 LITERATURE REVIEW	15
CHAPTER 3 METHODOLOGY	18
3.1 IMPLEMENTATION MODEL.....	18
3.2 TOOLS	20
3.3 DATA MODEL	23
CHAPTER 4 RESULTS AND DISCUSSION	24
4.1 SCREEN SHOT	24
4.2 TRANSFORMATION RULES	25
4.3 STEPS	26
4.4 ALGORITHM	27
4.4 TESTING	28
CHAPTER 5 CONCLUSION AND RECOMMENDATION	35
5.1 RELEVANCY TO THE OBJECTIVES.....	35

5.2	SUGGESTION FOR FUTURE WORK EXPANSION AND	
CONTINUATION		36
REFERENCES		38
APPENDICES		40
APPENDIX A CLAWS C5 TAGSETS		41
APPENDIX B CLAWS C7 TAGSETS		43

LIST OF FIGURES

Figure 1.1 Tagging Algorithm	10
Figure 3.1 Waterfall Model	18
Figure 3.2: JTagger Corpus.....	21
Figure 3.3: Penn Treebank Tagsets.....	22
Figure 3.2: Sequence Diagram of JTagger.....	23
Figure 4.1: JTagger Screen Shot	24
Figure 4.2: Process flow	27
Figure 4.3: Black-box Testing	28
Figure 4.4: JTagger Output	29
Figure 4.5: Monty Tagger Output for Sentence 1	31
Figure 4.6: JTagger Output for Sentence 1	31
Figure 4.7: Monty Tagger Output for Sentence 2	32
Figure 4.8: JTagger Output for Sentence 2	32
Figure 4.9: Performance Graph	34

LIST OF TABLES

Table 1.1 Open Word Classes	3
Table 3.1 Closed Word Classes	3
Table 4.1 Transformation Rules	25
Table 4.3 Table Performance.....	33

CHAPTER 1

INTRODUCTION

1.2 PART OF SPEECH

Linguists group the words of a language into classes (sets) which show similar syntactic behavior, and often a typical semantic type. These word classes are otherwise called syntactic or grammatical categories, but more commonly still by the traditional name part of speech (POS). Three important parts of speech are noun, verb, and adjective. Nouns typically refer to people, animals, concepts and things. The typical verb is used to express the action in a sentence. Adjectives describe properties of nouns. The most basic test for words belonging to the same class is the substitution test.

Example:

1. Children eat sweet candy.

The noun '*children*' refer to a group of people (those of young age) and the noun '*candy*' refers to particular type of food. The verb '*eat*' describes what children do with the candy.

Traditionally systems of parts of speech distinguish about 8 categories, but corpus linguists normally want to use more fine-grained classification of word classes. There are well-established sets of abbreviations for naming these classes, usually referred to as POS tags.

1.2 WORD CLASSES

According to Jurafsky [5], part-of-speech for English is divided into two large categories which are open class type and closed class type.

Word classes are normally divided into two. The open and lexical categories are ones like nouns, verbs and adjectives which have a large number of members and to which new words are commonly added. The closed or functional categories are categories such as prepositions and determiners (containing words like of, on, the) which have only a few members, and various parts of speech for a word are listed in an online dictionary, otherwise known as lexicon.

An **open word class** in linguistics is a word class that accepts the addition of new items through such processes as compounding, derivation, coining, or borrowing. Typical open word classes are nouns verbs and adjectives.

Open-class words are not considered part of the core language and as such they can be changed replaced or dropped from the common lexicon which can encompass many thousands of them. In English, words that belong to the open class type include the following parts of speech:

- Nouns
- Main verbs (not auxiliary verbs)
- Adjectives
- Adverbs
- Interjections

A **closed word class** in linguistics is a word class to which no new items can normally be added and that usually contains a relatively small number of items. Closed word class is always relatively few and resistant to change. Typical closed classes found in many languages are:

- Prepositions

- Determiners
- Conjunctions
- Pronouns

These tables illustrate the two kinds of word more clearly.

Open Word Classes			
Noun	Verb	Adjective	Adverb
Abstract: fear, joy	Transitive: bite, steal	Descriptive: lazy, tall	Manner: reluctantly, keenly, easily, softly
Concrete: chair, mud	Intransitive: live, cry	Comparative: lazier	Time: soon, often
Common: boy, town	Modal: can, will, may	Superlative: tallest	Place: here, there
Proper: Fred, Hull	Auxiliary: be, have, do		

Table 1.1 Open Word Classes

Closed Word Classes			
Determiner	Pronoun	Preposition	Conjunction
A, the, any, my, those, which	She, them, who, that, himself	In, across, at, by, near, within	And, but, if, or, while, unless

Table 1.2 Open Word Classes

1.2.1 Noun

Nouns are typically refers to entities in the world like people, animals and things. The general spelling endings of the plural, the genitive and the combined plural and genitive are the -s, -'s, and -s' endings or suffixes. A suffix is an affix that occurs at the end of the word; a prefix is one that occurs at the beginning.).

Examples are:

Dog, tree, person, hat, speech, idea

Nouns are traditionally grouped into proper nouns and common nouns. Proper nouns, like Regina, and IBM, are specific names or entities. In written English, proper nouns are usually capitalized. The names of days of week, months, institutions, organizations are proper nouns. A proper noun is the opposite of a common noun.

Examples of proper nouns:

1. *UTP* is located in *Tronoh, Perak*.
2. I have to go to the hospital on *Monday*.

A common noun is noun referring to a person, place, or thing in a general and usually in a capital letter only when it begins a new sentence. A common noun is the opposite of a proper noun.

Example of common nouns:

1. She told him that the *train* arrived at noon.
2. According to the *sign*, the nearest *town* is 60 *miles* away.

In many languages, including English, common nouns are divided into count nouns and mass nouns. Count nouns are those that allow grammatical enumeration; that is, they occur in both the singular and plural and they can be counted.

Example of count nouns:

1. Yesterday, Ali bought *one cat*.
2. The *relationship* between the classes is superclass and subclass.

Mass nouns are used when something is conceptualized as a homogeneous group. It refers to the noun which does not have a plural form, and which refers to something that we could or usually not counted. A non-countable noun always takes a singular verb in

a sentence. Non-countable nouns are familiar to collective nouns, and are the opposite of countable nouns.

Example of mass nouns:

1. Joseph Priestly discovered *oxygen*.
2. The *rain* is falling so heavily.

1.2.2 Pronoun

Pronouns are separated small classes of words that act like variables in that they refer to a person or entities. Pronouns like 'he', 'which' and 'you' can be used to make sentences less cumbersome and less repetitive. There are several types of pronouns which are personal pronoun, possessive pronoun and Wh-pronouns.

A personal pronoun refers to a specific person or thing and changes its form to indicate person, number, gender, and case.

Example of personal pronoun:

1. *She* is the only daughter in the family.
2. *I* have my own reason for not telling the truth.

Possessive pronouns are forms of personal pronouns that indicate either actual possession or more than often just an abstract relation between the person and some object. Some of the examples of possessive pronouns are 'mine', 'his' and 'ours'.

Example of possessive pronoun:

1. The book is belongs to *his* father.
2. *Our* main purpose for the project is to eliminate the communication cost.

The interrogative pronouns or Wh-pronouns are the same as the relative pronouns. It is used in certain question forms, or may also act as complement. '*whose*', '*which*' and '*what*' may also be used as determiners.

Example of interrogative pronouns:

1. *Who* came in?
2. *Whom* do you want?
3. *Whose* pens are these?

1.2.3 Verb

Verbs are used to describe actions, activities and states of being. The verbs are perhaps the most important part of a sentence.

Example of verb:

1. She *threw* the stone.
2. Mohammad *walked* along the river.

1.2.4 Adjective

Adjective is another inflected word class. An adjective modify a noun or a pronoun by describing, identifying, or qualifying words. It also includes many terms that describe properties or qualities. An adjective usually precedes the noun or the pronoun which it modifies.

Example of adjectives:

1. He is *more open* about it than she is.
2. They were *quite dead*.

1.2.5 Adverb

A typical adverb may be recognized by the '-ly' suffix that has been attached to an adjective, which most of them must be identified by untangling the grammatical relationships within the sentence or clauses as a whole. Unlike adjective, and adverb can be found in various places within a sentence.

Example of adverb:

1. She is a *friendly* person.
2. Hold it *closely* to you.

1.2.6 Preposition

Prepositions occur before noun phrases; semantically they are relational, often indicating spatial or temporal relations. Examples of preposition are 'in', 'on', 'about' and 'during'. Usually, preposition is followed by a noun.

Example of preposition:

1. The plane *took off* at 8am.
2. It is time to *take more* responsibilities.

1.2.7 Determiner

Nouns are often preceded by the words 'the', 'a', and 'an'. These words are called determiners. They indicate the kind of reference which the noun has.

Example of determiner:

1. *Those* apples are from grandma.
2. *A* taxi will be here in 10 minutes.

1.3 PART OF SPEECH TAGGING

Part-of-speech tagging is the process of marking up the words in a text with their corresponding parts of speech. People commonly learn a simplified form of this in their early years of school, identifying nouns, verbs, and so on. Tags play an important role in Natural Language applications like speech recognition, natural language parsing, information retrieval and information extraction. This is usually taken to be the first step in automatically processing language at the sentence level.

A common first step of analysis is to perform automatic grammatical tagging for categories roughly to find its conventional part of speech. One of the most well-known tag sets is the Penn Treebank are used in this project. Penn Treebank consists of 42 tagsets.

There are a few reasons why Penn Treebank has been chosen to be the tag set for the system. Some of the reasons are:

- The Penn Treebank tag sets distinguishes 45 categories found in most "traditional" grammars, such as adjectives, articles, adverbs, conjunctions, determiners, nouns, verbs etc. Tags are also attached to major punctuation marks, indicating their function.
- Since it have a small amount of tagsets, it is much more understandable especially to new learners in English. Penn Treebank simplifies it tag sets according to the grammar. For example, VB is actually verb.
- The Penn Treebank tag set distinguishes 9 punctuation tags, while C5 from BNC Corpus only come out with only 4.

1.4 TAGSETS FOR ENGLISH

The previous section gave broad descriptions of the kinds of syntactic classes that English words fall into. This section fleshes out that sketch by describing the actual tagsets used in part-of-speech tagging, in preparation for the various tagging algorithms to be described in the following sections.

There are a small number of popular tagsets for English, many of which evolved from the 87-tag tagset used for the Brown corpus. The Brown corpus is a 1 million word collection of samples from 500 written texts from different genres (newspaper, novels, non-fiction, academic, etc.) which was assembled at Brown University This corpus was tagged with parts-of-speech by first applying the TAGGIT program and then hand correcting the tags.

The other most commercial corpus that have been using in linguist area is the BNC corpus. The British National Corpus (BNC) is a 100 million word collection of samples of written and spoken language from a wide range of sources, designed to represent a wide cross-section of British English from the later part of the 20th century, both spoken and written. But both of the Brown Corpus and BNC Corpus are not freely available. It would cost about 500 pounds. Therefore, I have taken the alternative to collect words in the JTagger by manually tag the sentence or the one reused the one that is using in Brill Tagger and Monty Tagger.

The two of the most commonly used tagsets are the small 45-tag Penn Treebank tagset and the medium sized consists of 61 tag C5 tagset used by the Lancaster UCREL project's CLAWS (the Constituent Likelihood Automatic Word-tagging System) tagger to tag the British National Corpus (BNC).

The Penn Treebank tagset, which is used in this project, has been applied to the Brown corpus, the Wall Street Journal corpus, and perhaps partly because of its small size, it is one of the most widely used tagsets. Here are some examples of tagged sentences from the Penn Treebank version of the Brown corpus (I will represent a tagged word by placing the tag after each word, delimited by a slash):

- The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

1.5 BACKGROUND OF STUDY

Word sense disambiguation is the problem of assigning a sense to an ambiguous word by using its context. Word sense disambiguation means distinguishing words with the same spelling or pronunciation. Ambiguity is a pervasive phenomenon in human languages. It is very hard to find words that are not at least two ways ambiguous.

The two major approaches to this task are rule-based approaches and stochastic or statistical approaches. There is the third tagging algorithm which is transformation-

based tagging. Rule-based tagging is a method which learns a set of rules automatically based on a given corpus and then tags words following these rules [1]. The stochastic approaches computes probabilities of co-occurrence of words based on a given tagged corpus and then tags texts using these probabilities. Transformation-based tagging approached to machine learning, and draws inspiration from both the rule-based and the stochastic taggers.

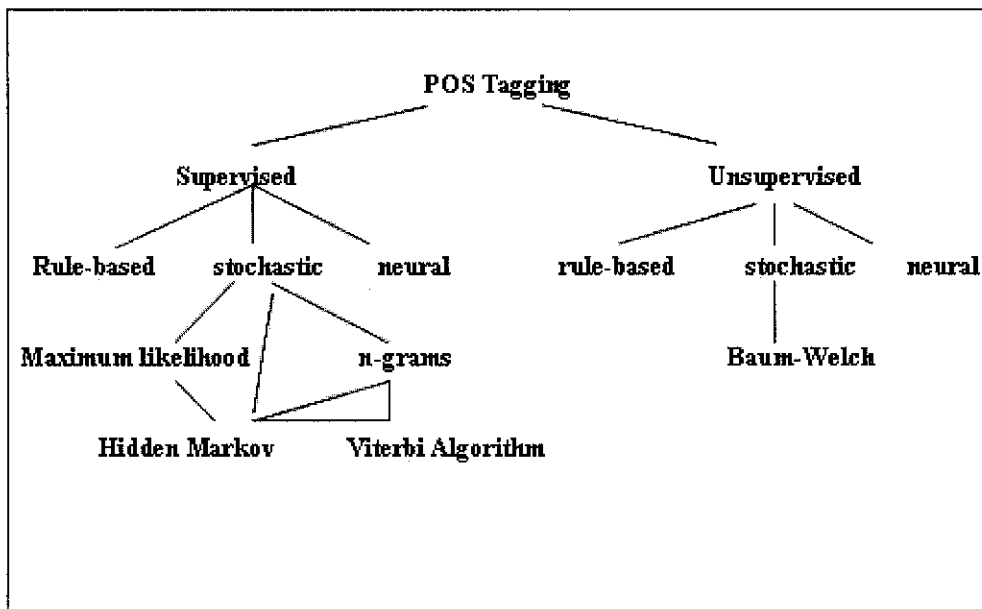


Figure 1.1: Tagging Algorithm

Part-of-speech tagging can be divided into two categories which are supervised and unsupervised (Figure 1). Supervised taggers typically rely on pre-tagged corpora to serve as the basis for creating any tools to be used throughout the tagging process. Unsupervised models, on the other hand, are those which do not require a pre-tagged corpus but instead use sophisticated computational methods to automatically induce word groupings [3,4].

1.6 PROBLEM STATEMENT

1.6.1 Word Sense Disambiguation

Part-of-speech tagging is very difficult because it deals with the ambiguities. In this case, one word could have different meaning. For example, "dogs" which is usually thought of as a just a plural noun, can also be a verb.

Schools commonly teach that there are eight parts of speech in English: noun, verb, adjective, preposition, pronoun, adverb, conjunction, and interjection. However, there are clearly many more categories and sub-categories. For example, adjectives divide into sub-classes for color, size, number, and other types of properties.

Examples of ambiguous word or sentence:

1. 'Bank'

- One meaning refer to the bank where the money transaction happened, but another one referred to the river bank.

2. 'Content'

- The word 'content' can be a noun or adjective. They are pronounced differently. The noun is pronounced *CONtent* and the adjective is *conTENT*.

3. 'The three big red dogs' and 'The red three big dogs'

- '*The three big red dogs*' is grammatical, but '*The red three big dogs*' is not. For nouns, plural, possessive, and singular forms can be distinguished. In many languages words are also marked for their case, grammatical gender, and so on; while verbs are marked for tense, aspect, and other things.

1.7 OBJECTIVES

The objectives of this project are as follows:

- **To help users to understand English better (POS)**

In the scope of part-of-speech (POS) tagging, one user can fully utilize the use of JTagger so that they will understand English better. JTagger can help the user to know which category the word belongs to depends on the structure of the sentence. Moreover, by using JTagger it could help to distinguish the ambiguous words and the tag set it belongs to.

- **To demonstrate the tagging method computationally**

The second objective is achieved at the end of the implementation of this system. In this case, the implementation of JTagger will help the user to handle mostly on the problem of disambiguation of words. The research focuses on the development of context-dependent grammar based on the Penn Treebank tagset. In addition, the system will use the rule-based approach since it is believed that the accuracy of the tagging method is much higher than stochastic tagging.

1.8 SCOPE OF STUDY

Part-of-speech (POS) tagging is an important step in natural language processing because identically written words may have different meanings. Part-of-speech tagging is the procedure during which the correct tag for an ambiguous word is selected. Computer programs able to do the process automatically are called part-of-speech taggers.

In this project, JTagger is capable to tag a sentence by using rule-based tagging algorithm based on Penn Treebank Tagsets. The project developed a medium sized learning corpus for English only. For the time being, the corpus consists of

approximately about 200,000 words. Since the available corpus like BNC needed to be paid, I have taken an initiative to manually tag the words and categories to its tag sets. Other from that, I also took some already tagged sentence from existing program available in the web such as from Brill Tagger and also Monty Tagger. Both of the taggers are using Penn Treebank which is the same as the JTagger.

The first semester is more emphasized on the research of the Natural Language Processing itself and the second semester is the development phase. I have managed to finish the project within the time frame. Since the research and development in 2 semesters in other ways is in one year, it really given beneficial to me so that more research and testing can be done. The time frame for the whole project is attached in the appendices.

1.9 SIGNIFICANT OF PROJECT

- **Help to Understand English better**

By using JTagger, one could understand English better and used the system to tag a sentence to know the meaning of the sentence and know which class of the word belongs to. For nouns, plural, possessive, and singular forms can be distinguished by using the tagger.

- **Information about the word and its neighbors**

Part-of-speech tagging in language processing gives a significant amount of information about the word and its neighbors. These tagsets distinguished between possessive pronoun like *my*, *your*, *his*, *her* and *its* and personal pronouns like *I*, *you*, *he*, and *she* [2]. Possessive pronouns are likely to be followed by a noun, personal pronouns by a verb. This can be useful in a language model for speech recognition or speech tagging.

- **Reveal the word pronunciation**

Another benefits from using part-of-speech tagging, it will reveal about how the word is pronounced. Taking the word 'discount' into consideration, it could be noun or pronoun. '*DIScount*' is pronounced as noun and '*disCOUNT*' is a verb. *DIScount* means giving reduction but *disCOUNT* means miscalculation. Thus knowing the part-of-speech can produce more natural pronunciations in a speech synthesis system and more accurate in a speech tagging.

CHAPTER 2

LITERATURE REVIEW

According to Roberts [1], “*machine learning is concerned with acquiring knowledge from an environment in a computational manner, in order to improve the performance*”. By speech and language processing, those computational techniques that are process of spoken and written in human language. The difference between these language processing applications from other data processing is their use of knowledge. The process that involves in the machine learning is speech recognition and speech tagging.

Dolan [2] mentioned that, “*the problem of word sense disambiguation is one which has received increased attention in recent work on Natural Language Processing (NLP) and Information Retrieval (IR)*”. The problem in word sense ambiguity is that many words have several meanings or sense. For such words, there is thus ambiguity about how they are to be interpreted. The task of disambiguation is to determine which of the sense of an ambiguous word is invoked in a particular use of the word. This is done by looking at the context of the word use.

Guilder [3] in her article mentioned that there are two type of part of speech tagging. She said that, “*One of the first distinctions which can be made among the POS taggers is in term of the degree of automation of the training and tagging process. The terms commonly applied to this distinction are supervised vs. unsupervised.*” Basically, supervised learning is when we know the actual status for each piece of data that we

train, whereas with the unsupervised learning, we do not know the classification of the data in the training example.

Part-of-speech tagging plays an important role in many areas of natural language processing. Brill [4] said that, *the main purpose of using rule-based approach is because it is believed that it could give better accuracy than stochastic rule*. Rule-based system learns a set of rules automatically based on a given corpus and then tags words following these rules. According to Brill [4] *“Stochastic tagger have obtained a high degree of accuracy without performing any syntactic analysis on the input. The stochastic part of speech taggers make use of a Markov model which captures lexical and contextual information. Once the parameters of the model are estimated, a sentence can then be automatically tagged by assigning it the tag sequence which is assigned the highest probability by the model.”*

A stochastic model of the type described above may work well, as similar sequences of text will be found easily, and these will have similar tags assigned to them. However, if the similarity between dictionary entries is too great, this may lead to the common over fitting problem or not enough variance in the data (particularly the training data) will not allow the words to be tagged very efficiently. From this, it seems that a rule-based approach may well work better than a probabilistic one.

Brill[4] also pointed out that the rule-based tagger *“has many advantages over these taggers, including: a vast reduction in stored information required, the perspicuity of a small set of meaningful rules, ease of finding and implementing improvements to the tagger, and better portability from one tag set, corpus genre or language to another.”* Overheads for programs tend to increase in relation to the size of the file being parsed. The dictionary is a very large text file, so approaches that require large amounts of extra memory or disk space would be harder to run, especially if it had to be run on a remote, larger computer. This would make it extremely cumbersome to run tests and alter the program if necessary. A small set of meaningful rules would also make it easier to alter the tagger to improve its performance. The smaller the tag set, the higher the accuracy.

However, a very small tag set tends to make the tagging system less useful since it provides less information. So, there is a drawback here. Another issue in tag-set design is the consistency of the tagging system. Words of the same meaning and same functions should be tagged with the same tags.

According to Lee. G, [10] *“Both statistical and rule-based approaches to part-of-speech (POS) disambiguation have their own advantages and limitations. Especially for Korean, the narrow windows provided by hidden markov model (HMM) cannot cover the necessary lexical and long- distance dependencies for POS disambiguation. On the other hand, the rule-based approaches are not accurate and flexible to new tag-sets and languages.”* JTagger is using the rule-based method to eliminate the hand-tagging. Although the rule-based also have its disadvantages and limitation, but it is easier to develop and manage. Stochastic on the other hand, are developed using formula and calculation. JTagger can also eliminate the storage used since it only used for storing the rules and no calculation is needed.

CHAPTER 3

METHODOLOGY

The methodology that is used in the project is the Waterfall Model. The main reason why I chose this model is because it adopted a formal step-by-step approach to the System Development Life Cycle (SDLC) that moves logically from one phase to the next. Phases of this methodology are as follow:-

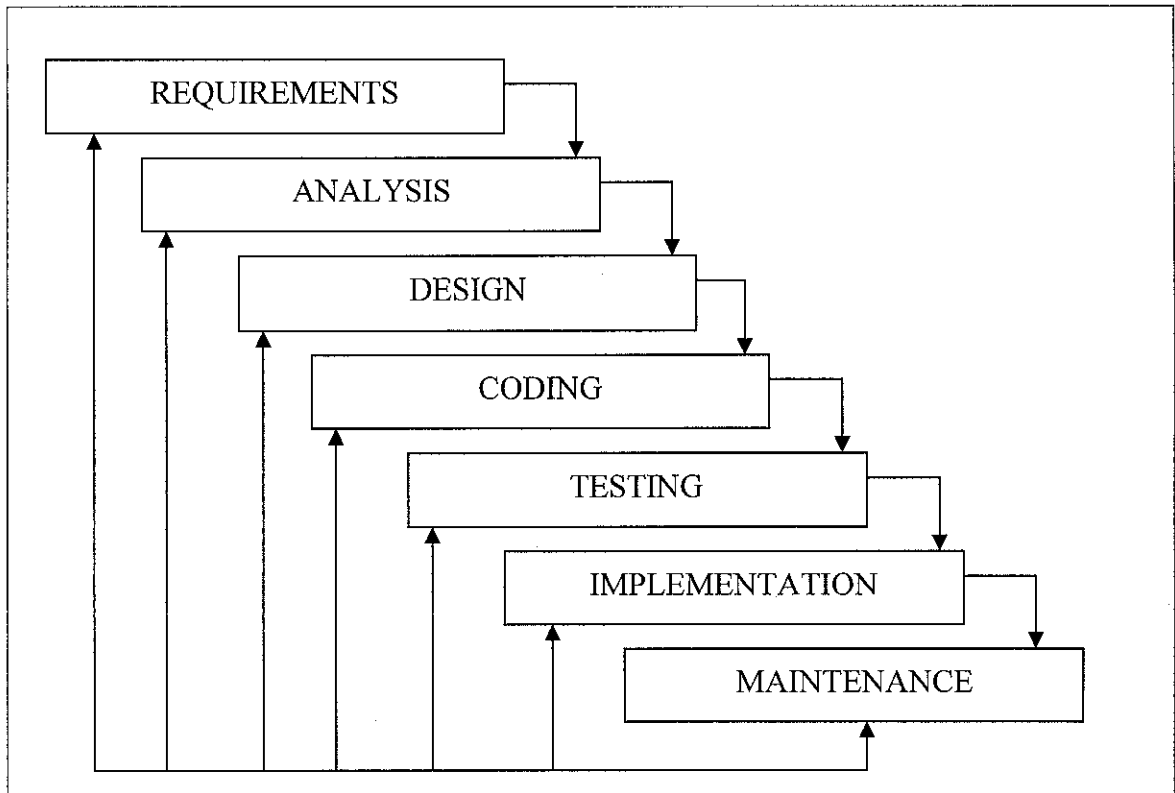


Figure 3.1: Waterfall Model

- **Requirement Planning Phase**

At this phase, identifying and analyzing the project requirements is done. The research is focused on the Natural Language Processing application. The main area that concerned is on the part-of-speech tagging for the rule-based algorithm. All the important attributes and key words are identified and used for the whole development. Finally it goes to the recognition of tools such as hardware and software which are also being identified and have been list down later in this section.

- **Analysis Phase**

In the analysis phase, all the steps are identified and measured so that it met the scope of the project. For example, all the rules that have been developed will be analyzed to make sure it will give the most reasonable accuracy for the system.

- **Design Phase**

System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model. In the design phase, the interface of the system is designed to meet the specific requirements.

- **Coding Phase**

This is the phase where design is translated into programming language. First, the corpus which is used as a dictionary to tag the initial sentence is developed. The focus of this phase is on how to develop the system that will tag a sentence and give a reasonable accuracy to the user. The tool used is JAVA programming language.

- **Testing**

The testing phase came after all the phases above are completed. In this phase, the system is tested by entering the sentence and check whether it is given the correct tag set for each of the words. Later, the words will retag using the rules that have been

developed.

- **Implementation**

In the implementation phase, the system is presented and ready to be used.

- **Maintenance**

During the maintenance phase, all the errors are fixed and any problem is required to solve.

3.1 TOOLS

In preceding this project, below listed the software and hardware that is used.

3.1.1 Software

- Programming tools (Java)

Advantages Using JAVA

The main reason why Java is chosen to be the programming tools in developing this project is because JAVA is an independent platform. It can run in any operating system including LINUX or Windows.

Java can do all the file manipulation and text searching while at the same time, it has all the graphical capabilities of a language like C.

Moreover, it is free. There are free Java implementations fore very type of computer. In addition, code written and compiled on one type of machine will run on any other type. This also suggests that Java programs can continuously be use for many years, since the language is widely used.

3.1.2 Corpus

In this project, I have manually tagged the sentence using the existing Monty Tagger to get the final result. Apart from that, I also has taken some initiative of taking the initial word with tag sets from the Monty Tagger itself. The total words in the corpus are approximately 3000 words.

Below is the example how the corpus look like:

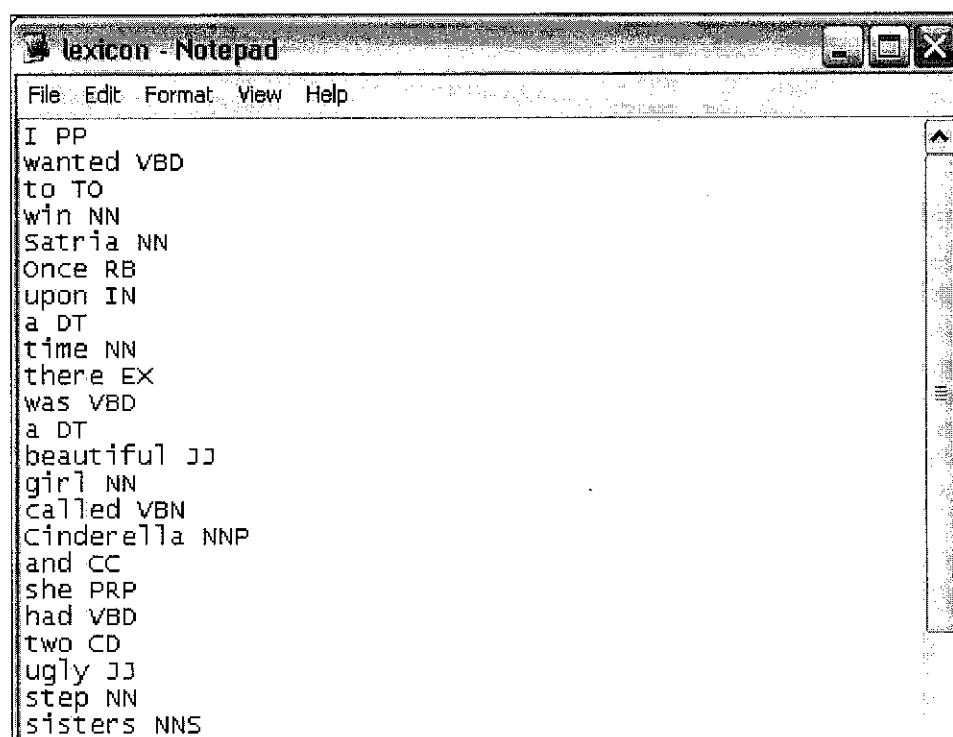


Figure 3.2: JTagger Corpus

Basically JTagger is a supervised learning tagger. It typically relies on pre-tagged corpus/corpora to serve as the basis for creating any tools to be used throughout the tagging process.

3.1.3 Tagsets

To take a sentence, the system required to use tagsets. The tagsets that I used is the Penn Treebank Tagsets. It consists of 45 tags which included the verb, noun,

preposition and other word classes. Penn Treebank is considered as a small tagsets compared to other Tagsets like C5 or C7 from BNC Corpus.

In the corpus, I have made a restriction in the tagset. One word should only belong to one category. The changing of tagsets happened after the rule is applied.

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	<i>and, but, or</i>	SYM	Symbol	<i>+, %, &</i>
CD	Cardinal number	<i>one, two, three</i>	TO	"to"	<i>to</i>
DT	Determiner	<i>a, the</i>	UH	Interjection	<i>ah, oops</i>
EX	Existential 'there'	<i>there</i>	VB	Verb, base form	<i>eat</i>
FW	Foreign word	<i>mea culpa</i>	VBD	Verb, past tense	<i>ate</i>
IN	Preposition/sub-conj	<i>of, in, by</i>	VBG	Verb, gerund	<i>eating</i>
JJ	Adjective	<i>yellow</i>	VBN	Verb, past participle	<i>eaten</i>
JJR	Adj., comparative	<i>bigger</i>	VBP	Verb, non-3sg pres	<i>eat</i>
JJS	Adj., superlative	<i>wildest</i>	VBZ	Verb, 3sg pres	<i>eats</i>
LS	List item marker	<i>1, 2, One</i>	WDT	Wh-determiner	<i>which, that</i>
MD	Modal	<i>can, should</i>	WP	Wh-pronoun	<i>what, who</i>
NN	Noun, sing. or mass	<i>llama</i>	WP\$	Possessive wh-	<i>whose</i>
NNS	Noun, plural	<i>llamas</i>	WRB	Wh-adverb	<i>how, where</i>
NNP	Proper noun, singular	<i>IBM</i>	\$	Dollar sign	<i>\$</i>
NNPS	Proper noun, plural	<i>Carolinas</i>	#	Pound sign	<i>#</i>
PDT	Predeterminer	<i>all, both</i>	"	Left quote	<i>(' or ")</i>
POS	Possessive ending	<i>'s</i>	"	Right quote	<i>(' or ")</i>
PP	Personal pronoun	<i>I, you, he</i>	(Left parenthesis	<i>([({ <</i>
PPS	Possessive pronoun	<i>your, one's</i>)	Right parenthesis	<i>(]) } ></i>
RB	Adverb	<i>quickly, never</i>	,	Comma	<i>,</i>
RBR	Adverb, comparative	<i>faster</i>	.	Sentence-final punc	<i>(. ! ?)</i>
RBS	Adverb, superlative	<i>fastest</i>	:	Mid-sentence punc	<i>(: ; ... - -)</i>
RP	Particle	<i>up, off</i>			

Figure 3.3: Penn Treebank Tagsets

Many word tokens are unambiguous, and so will be assigned just one tag: e.g. *various* AJ0 (adjective).

To find the list of potential tags associated with a word, JTagger first looks up the word in a lexicon of 3,000 word entries in the corpus mentioned above. This lexicon look-up accounts for a large proportion of the word tokens in a text file.

However, for any rarer words or names will not be found in the lexicon, they are tagged by other test procedures. Some of the other procedures are:

- Look for the ending of a word: e.g. words in *-ness* will normally be nouns.

- Look for an initial capital letter (especially when the word is not sentence-initial). Rare names which are not in the lexicon and do not match other procedures will normally be recognized as proper nouns on the basis of the initial capital.
- Look for a final *-(e)s*. This is stripped off, to see if the word otherwise matches a noun or verb; if it does, the word in *-s* is tagged as a plural noun or a singular present-tense verb.
- If all else fails, a word is tagged ambiguously as a noun, an adjective or a lexical verb.

3.2 DATA MODEL

The procedure of using JTagger is simple. The users only have to enter the sentence that they wanted to tag. There are some limitation that required which is user is only allowed to enter only one sentence. But, there's no limitation in the number of words in the sentence. After the sentence is entered, the user will have to click on the "Tag Now" button and the result will be displayed in the result box. All the backend procedure in changing the tag sets according to their tag set will be not shown. The sequence diagram below will explain how JTagger works and the procedure that I used.

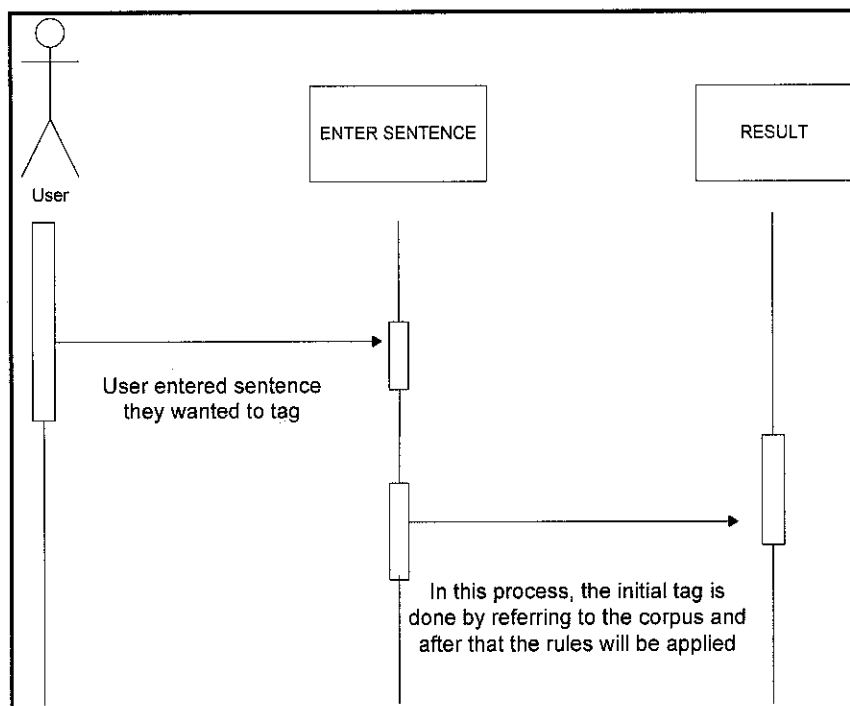


Figure 3.2: Sequence Diagram of JTagger

CHAPTER 4

RESULT AND DISCUSSION

4.1 SCREEN SHOT

The screen design for JTagger is as follow:

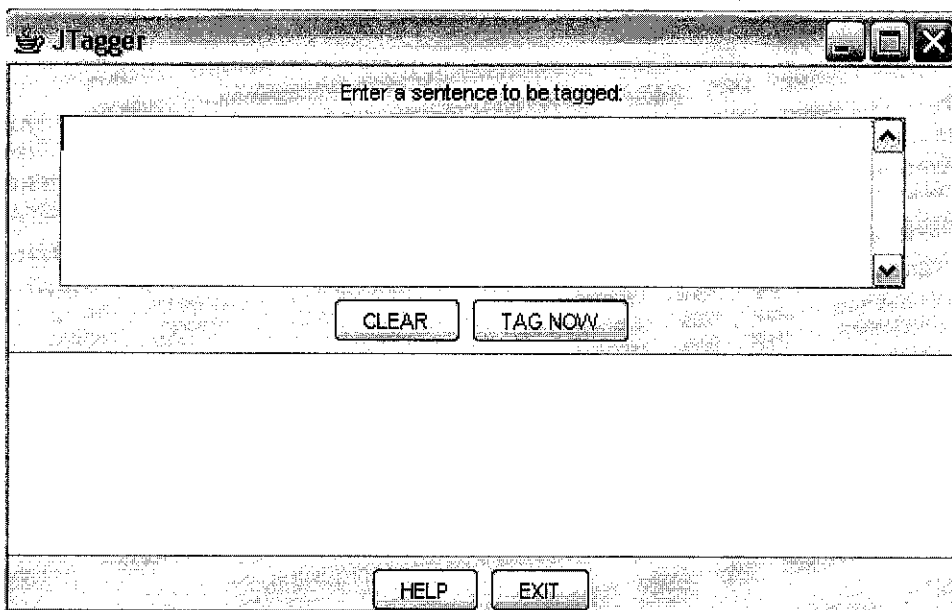


Figure 4.1: JTagger Screen Design

- **Input field:** User will enter the sentence that they want to tag.
- **Button Tag Now:** The function of the button is for the user to choose when they wanted to tag the sentence.
- **Button Help:** The function of the help button is to guide the user on how to use the system.
- **Button Reset Form:** The function of this button is to clear the input field and the output field so that user can enter a new sentence that they wanted to tag.
- **Exit:** Button exit is used when the user wanted to exit from the system.

- **Basic Idea of JTagger**

JTagger tagged the most probable tag for each value inserted in the input field. The name JTagger is applied since the system is basically using Java as its tools for developing interface and source codes. JTagger is using a small, manually and correctly annotated corpus - the training corpus - which serves as input to the tagger. The system derived the information from the training corpus and then applies it to the most likely part of speech tag for a word. Once the training is completed, the tagger can be used based on the tagset of the training corpus.

The speed of the tagging depended on the capacity of the corpus. The lexicon file contains the frequencies of each word found in the manually hand tagged corpus. It changed tags according to rules of type “if word-1 is a determiner and word is a verb then change the tag to noun” in a specific order.

Basically, JTagger labels each word with the most likely tagged.

For example:

- *race* has the following probabilities in the Brown corpus:

- $P(NN|race) = .98$
- $P(VB|race) = .02$

Transformation rules make changes to tags

- “Change NN to VB when previous tag is TO”
... is/VBZ expected/VBN to/TO race/NN tomorrow/NN
 becomes
... is/VBZ expected/VBN to/TO race/VB tomorrow/NN

4.2 TRANSFORMATION RULES

No.	Change Tag		Condition
	From	To	
1	NN	VB	Previous tag is TO

2	VBP	VB	One of the previous three tag is MD
3	NN	VB	One of the previous two tag is MD
4	VB	NN	One of the previous two tag is DT
5	VBD	VBN	One of the previous three tag is VBZ
6	VBN	VBD	Previous tag is PRP
7	VBN	VBD	Previous tag is NNP
8	VBD	VBN	Previous tag is VBD
9	VBP	VB	Previous tag is TO
10	POS	VBZ	Previous tag is PRP
11	VB	VBP	Previous tag is NNS
12	VBD	VBN	One of the previous three tag is VBP
13	IN	WDT	One of the next two tags is VB
14	VBD	VBN	One of the previous two tag is VB
15	VB	VBP	Previous tag is PRP
16	IN	WDT	Next tag is VBZ
17	IN	DT	Next tag is NN
18	JJ	NNP	Next tag is NNP
19	IN	WDT	Next tag is VBD
20	JJR	RBR	Next tag is JJ

Table 4.1 Set of Tagging Rules

4.3 STEPS

Steps

- Step 1: Initially map words with the most common tag stored in the database
- Step 2: Apply transformation rules to the initially tagged words
- Step 3: Retagged words by applying the rules
- RESULT: The newly tagged sentence

4.4 ALGORITHM

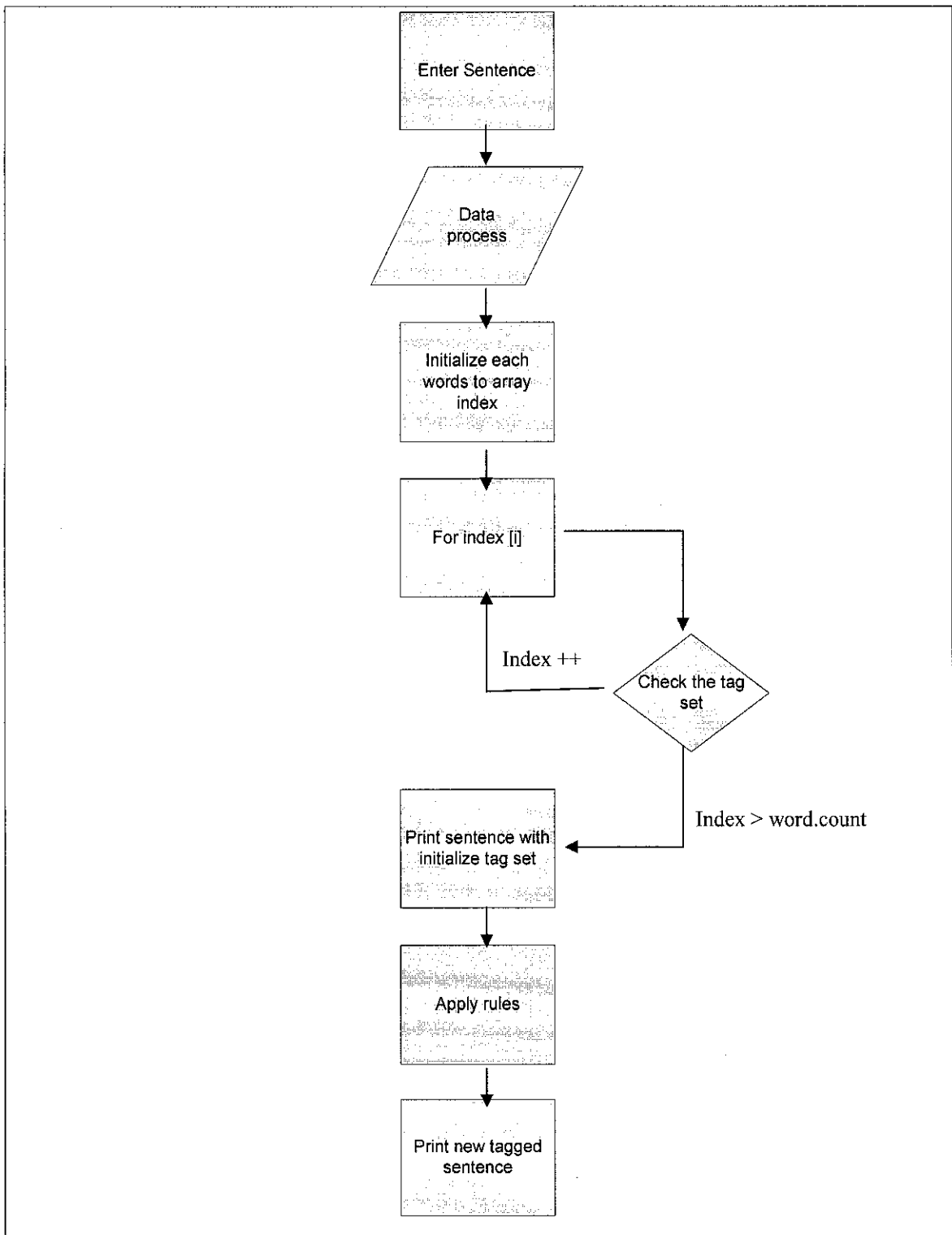


Figure 4.2: Process flow

4.5 TESTING

Software testing is the process used to help identify the correctness, completeness, security and quality of developed system.

In this project, I used the black box testing as a method to determine the correctness and completeness of the system. Basically, black box testing is an approach to testing where the tests are derived from the program or component specification. JTagger will act as a 'black-box' whose behaviors can only be determined by studying its inputs and the related outputs.

Below is the diagram that could help to understand more on the method.

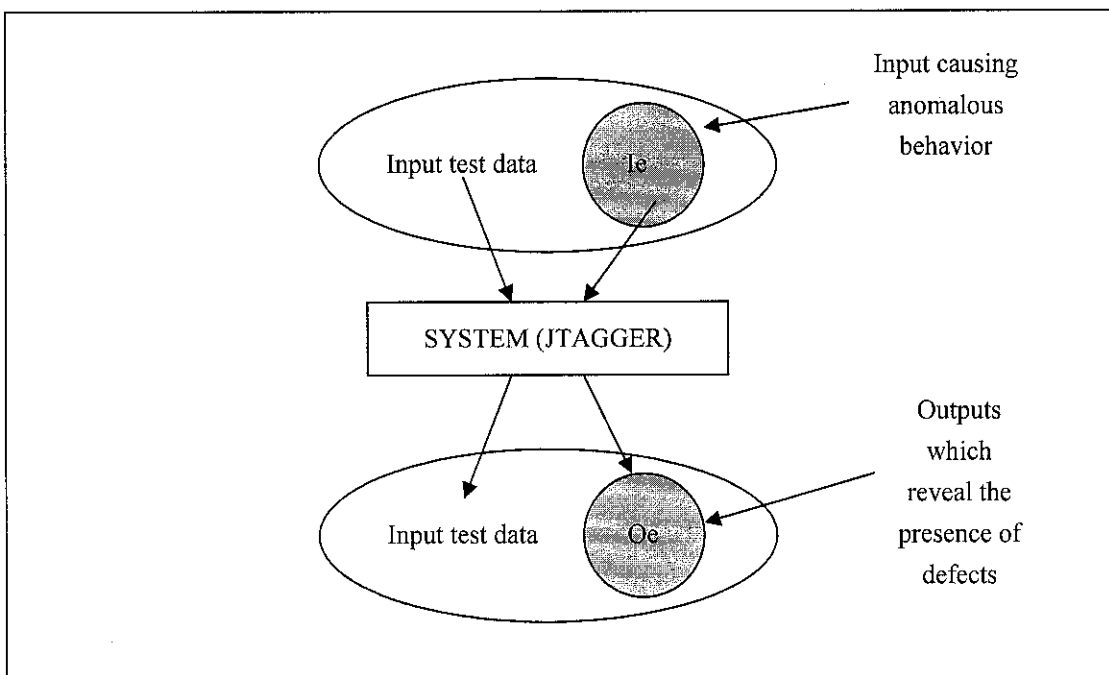


Figure 4.3: Black-box Testing

The advantages of this type of testing include:

- The test is unbiased because the designer and the tester are independent of each other.
- The tester does not need knowledge of any specific programming languages.
- The test is done from the point of view of the user, not the designer.

- Test cases can be designed as soon as the specifications are complete.

RESULT 1

For the JTagger, I have used the same techniques in testing the system. A list of sentence that suitable for the testing are tried. For example:

Input:

He is expected to race tomorrow.

Output:

He/NN is/VBZ expected/VBN to/TO race/VB tomorrow/NN.

The word 'race' initial tag in the corpus is actually noun (NN). But after applying the rules it became a verb since the rule is "Change NN to VB when previous tag is TO".

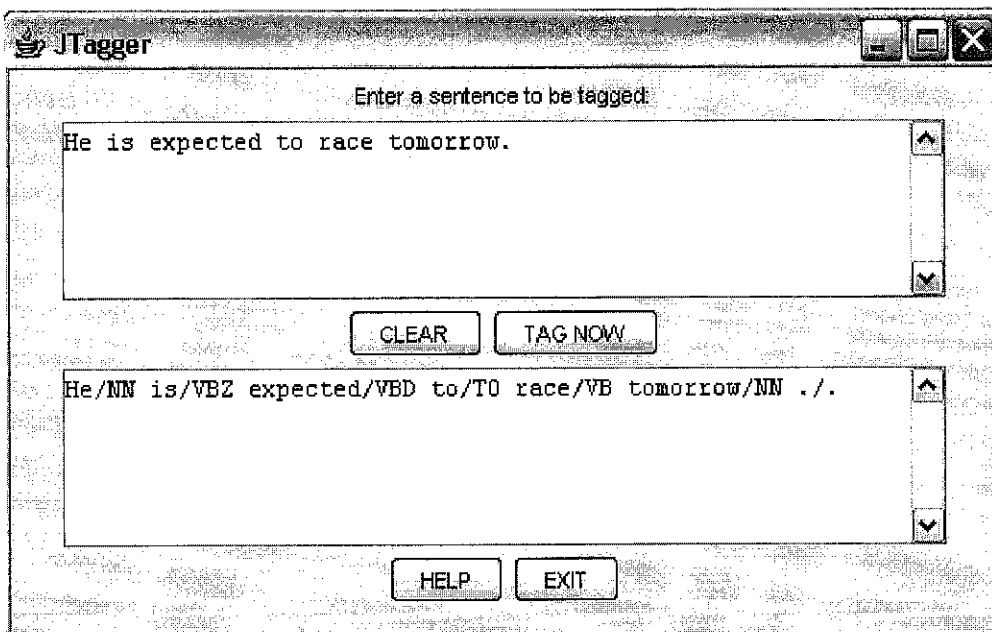


Figure 4.4: JTagger Output

RESULT 2

To get a more accurate result on the output of JTagger, I have taken an initiative to compare the output from JTagger with Monty Tagger. This is to know the accuracy level of the result that produced by JTagger.

Basically, Monty Tagger and JTagger are using the same tag sets which are the Penn Treebank tag sets. So in this case, it is much easier to compare both of them. Monty Tagger is a rule-based part-of-speech tagger based on Eric Brill's 1994 transformational-based learning POS tagger, and uses Brill-compatible lexicon and rule files. (The distribution includes Brill's original Penn Treebank trained lexicon and rule files.) It also includes a tokenizer for English and tools for performance evaluation.

The Monty Tagger is implemented using Python and JTagger used JAVA. The programming languages for both type of tagger are different but the approach is the same.

To make it clearer, I have chosen a similar sentence for the testing. Below is some the comparison that has been done.

Sentence:

1. He will race the car.
2. When will the race end?

```
C:\WINDOWS\system32\cmd.exe
F:\NYP Part1\MontyTagger\montytagger-1.2-java\montytagger-1.2-java>java -cp .;montytagger.jar montytagger.MontyTagger

##### INITIALIZING #####
Fast lexicon found! Now loading!
Lexicon OK!
LexicalRuleParser OK!
ContextualRuleParser OK!
#####

MontyTagger v1.0
--send bug reports to hugo@media.mit.edu--

> He will race the car.
He/PRP will/MD race/VB the/DT car/NN ./
-- monty took 0.31 seconds. --

>
```

Figure 4.5: Monty Tagger Output for Sentence 1

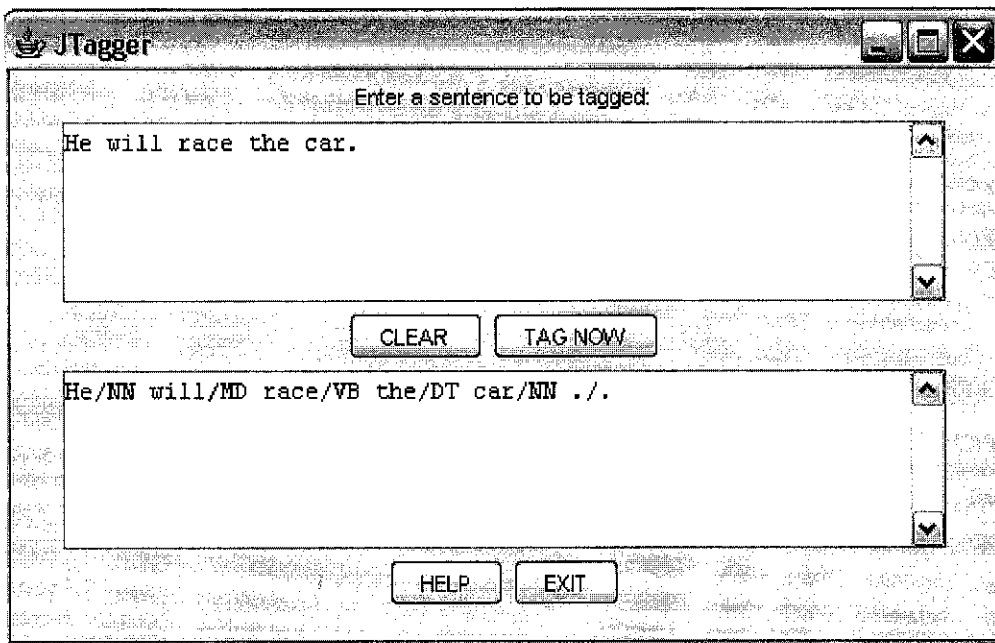


Figure 4.6: JTagger Output for Sentence 1


```
C:\WINDOWS\system32\cmd.exe
F:\NYP Part1\MontyTagger\montytagger-1.2-java\montytagger-1.2-java>java -cp .;montytagger.jar montytagger.MontyTagger

##### INITIALIZING #####
Fast Lexicon Found! Now Loading!
Lexicon OK!
LexicalRuleParser OK!
ContextualRuleParser OK!
#####

MontyTagger v1.0
--send bug reports to hugo@media.mit.edu--

> When will the race end?

When/WRB will/MD the/DT race/NN end/NN ?/.
-- monty took 0.3 seconds. --

>
```

Figure 4.7: Monty Tagger Output for Sentence 2

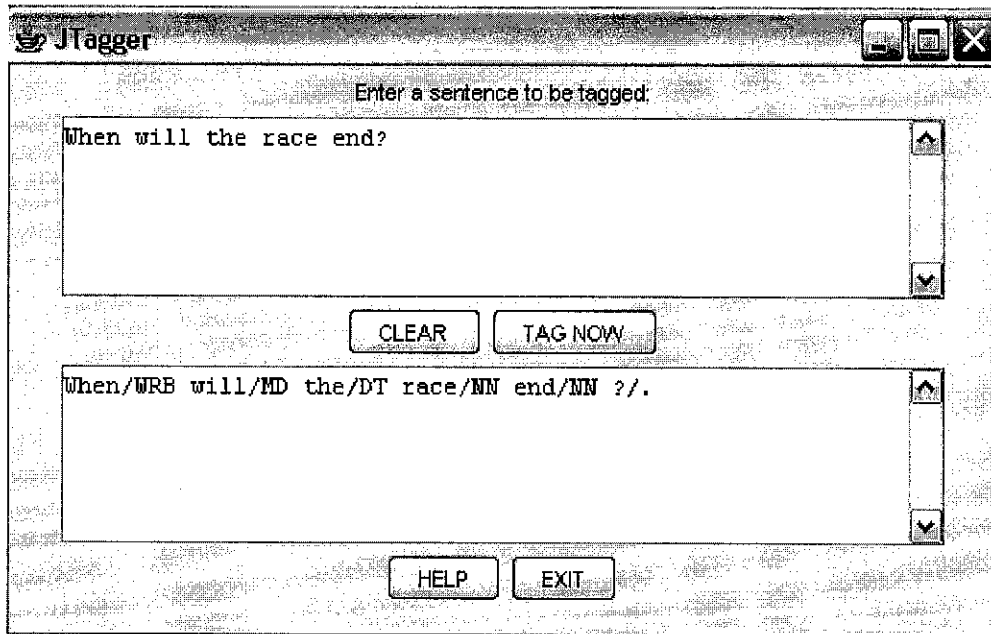


Figure 4.8: JTagger Output for Sentence 2

The results above show that the rule is successfully been implemented. The rule that used is:

“Change NN to VB when previous tag is TO”

The description below showed how race will become noun (NN) or verb (VB) according to the structure of the sentence. When the rule is applied, the word race will become Verb when the previous word is TO which is the determiner. Compared both of the results, the JTagger tagged the word “He” in the sentence of “He will race the car” as a noun instead of Monty Tagger which tagged it differently. This is because the initial tag sets applied to each of the tagger is different. This show that the result cannot be 100% accurate because it highly depends on the tag sets and the rules that applied. Since JTagger is depends solely on the rule based it cannot precisely determined the accuracy of the result.

RESULT 3

To measure the accuracy of JTagger, I have prepared fifty sentences that have ambiguous words. The results are then compared with Monty Tagger and from there I know how many percentage of accuracy the JTagger gives. To make it more efficient, I asked one user that expert in English to manually tag the sentence according to her understanding and used the Penn Treebank as reference tag sets.

The table below shows the performance of both taggers and manual tagged.

	Number of Sentence	Number of correct sentence(s) tagged	Number of wrong sentence(s) tagged	% of accuracy
JTagger	50	36	14	72%
Monty Tagger	50	43	7	86%
Manual Tagging	50	45	5	90%

Table 4.2: Table Performance

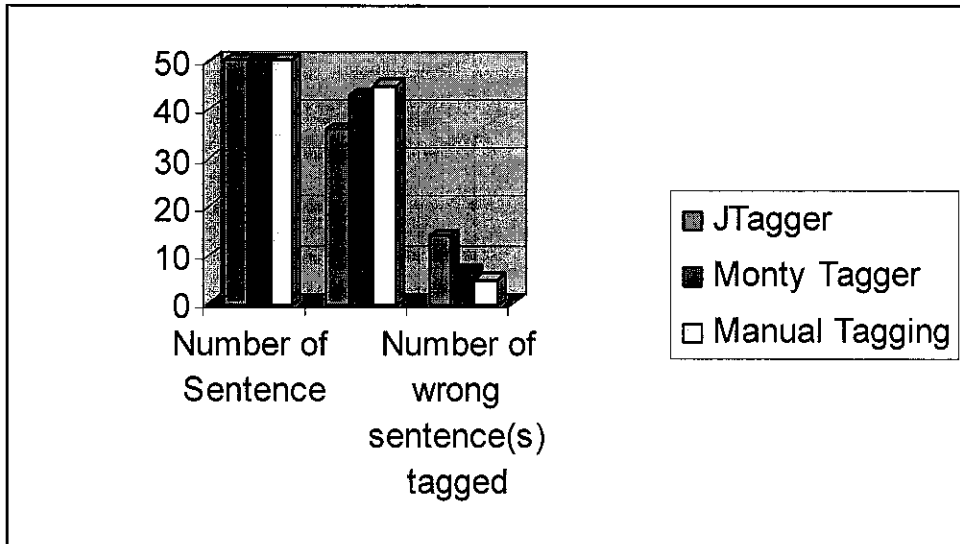


Figure 4.9: Performance Graph

Actually it is difficult to compare results with other published results. For JTagger, it can only produce 73% of accuracy from the test sentences. Monty tagger in the meanwhile can produce more accurate result. But the highest percentage would be for manually tagging. The manually tagging is simple but it depends on the ability of the user and their knowledge in English. Furthermore, it takes times to finish the test. JTagger and Monty Tagger are proven can fasten the manually tagging process.

Regardless of the precise rankings of both taggers, I have demonstrated that simple rule-based tagger. JTagger is proven that it served 72% of accuracy. While Monty Tagger produced more accurate results since it combines stochastic rules. It also has its own training set than serves as the patches to the tagger. While JTagger is 100% depends on the rules and also the corpus that serves as the input to initial tagging.

Incorporating a large corpus into the tagger would basically improve the performance as it would increase the error resulting from the initial tagging. Second, the accuracy of the results also depends on the tagging rules. The more rules are implemented, the better the results.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 RELEVANCY TO THE OBJECTIVES

The JTagger project which is how to determine the part of speech tagging in a sentence is very important where it automatically tagged the sentence using the rule-based method. The tagger works by automatically assigning each word to its most likely tag, and then applies the rule that have been developed.

Objectives one for this project is to help users to understand English better in the part of speech tagging context. The result proved that JTagger helped to understand English better and the user would know which category of word it belongs to depending on the structure of it. By applying the rule, one user can resolve the disambiguation of the word itself.

For example, *book* is ambiguous. That is, it has more than one possible usage and part-of-speech. It can be a verb (as in *book that flight* or *to book the suspect*) or a noun (as in *hand me that book*, or *a book of matches*). Similarly *that* can be a determiner (as in *Does that flight serve dinner*), or a complementizer (as in *I thought that your flight was earlier*). The problem of JTagger is to resolve these ambiguities, choosing the proper tag for the context. Part-of-speech tagging is thus one of the many disambiguation tasks as I have addressed earlier.

The second objective is to demonstrate the tagging method computationally. Before this, all the tagging processes were done manually hand-tagged. By implemented such a system likes JTagger, that could help eliminates the problems. The other matter addressed in the objectives was how JTagger as a machine learning translation and most importantly could help in word disambiguation. This is already achieved and is relevant to the objectives of the JTagger implementation in early beginning.

Both of the objectives for JTagger are relevant to the final results. The expected results is also achieved thus it would gives benefits to the users. Furthermore, JTagger have its commercial value to the English learners and users that could benefit from it.

5.2 SUGGESTION FOR FUTURE WORK EXPANSION AND CONTINUATION

Although JTagger is successfully implemented, but there's more things that can be done to make it more useful. For future work expansion, here's some of the suggestion that might be considered:

- **Online Web**

Now, JTagger is implemented as a stand-alone system. Future development for enhancement can be done for the online web for JTagger. This objective is important since it could help the user use the system online.

- **Searching Method**

JTagger is basically is using the sequential searching method which in this case, it will search the word in the corpus one by one. Future enhancement can also consider on changing the searching technique since more words in the corpus, the more time it will take for the execution. The suggested method is the binary search which is faster.

- **Use Commercial Corpus**

To produce a more accurate result, it is better if the corpus is able to come out with the initial tag. But, there the drawback of having a bigger corpus would result is slowing the system and also it would cost more.

- **Combine Rule-based with Stochastic (probability) Method**

JTagger is solely based on the rule-based method. To make it more efficient, it is better to combine both of the rule-based and stochastic method so that a more accurate result can be achieved. This means the combination of rule-based and probability of words in the sentence. Also future suggestion can also concentrate on developing a rule-s learning system which can serves as the input to the JTagger.

- **Variety of Language**

JTagger is an English based machine learning system. But, future work can expand to the other languages. There are now many of languages who adapt this kind of tagger for better understanding on the language. Perhaps, Bahasa Malaysia could be used as one the expansion of tagging system.

REFERENCES

- [1] Roberts, A., 2003. Machine Learning in Natural Language Processing.

- [2] Dolan, W. B. E. Word Sense Ambiguation : Clustering Related Sense. Microsoft Research.

- [3] Guilder, L. V, Automated Part of Speech Tagging : A Brief Overview. Georgetown University.

- [4] Brill, E. 1992. A Simple Rule-Based Part of Speech Tagger. Department of Computer Science, University of Pennsylvania.

- [5] Jurafsky, D. & Martin, J.H. 2000. Speech and Language Processing : An Introduction to Natural Language Processing. Prentice Hall.

- [6] Brill, E. 1995. Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging. Department of Computer Science, John Hopkins University.

- [7] Manning, C.D & Schutze, H 2001. Foundation of Statistical Natural Processing Language. Massachusetts Institute of Technology.

- [8] Wardhough, R. 2003. Understanding English Grammar: A Linguistic Approach. Blackwell Publishing.

- [9] Ambiguous words: <http://en.wikipedia.org/wiki/> retrieved from the World Wide Web on May 12, 2006.
- [10] Lee, G & Lee, J.H & Shin, S. TAKTAG: Two-phase learning method for hybrid statistical/rule-based part-of-speech disambiguation. Department of Computer Science & Engineering and Postech Information Research Laboratory, Pohang University of Science & Technology , Korea
- [11] Yu, L, Ahmed, S.T., Gonzalez, G, Logsdon, B, Nakamura, M, Nikkila, S, Shah, K, Tari, L, Wendt, R, Zeighler, A, & Baral, C 2005. Genomic Information Retrieval through Selective Extraction and Tagging by the ASU-BioAI Group, Department of Computer Science and Engineering, Arizona State University, United States of America.

APPENDICES

APPENDIX A

UCREL CLAWS C5 Tagset

!	PUN
"	PUQ
(PUL
)	PUR
,	PUN
-	PUN
.	PUN
...	PUN
:	PUN
;	PUN
?	PUN
AJ0	adjective (unmarked) (e.g. GOOD, OLD)
AJC	comparative adjective (e.g. BETTER, OLDER)
AJS	superlative adjective (e.g. BEST, OLDEST)
AT0	article (e.g. THE, A, AN)
AV0	adverb (unmarked) (e.g. OFTEN, WELL, LONGER, FURTHEST)
AVP	adverb particle (e.g. UP, OFF, OUT)
AVQ	wh-adverb (e.g. WHEN, HOW, WHY)
CJC	coordinating conjunction (e.g. AND, OR)
CJS	subordinating conjunction (e.g. ALTHOUGH, WHEN)
CJT	the conjunction THAT
CRD	cardinal numeral (e.g. 3, FIFTY-FIVE, 6609) (excl ONE)
DPS	possessive determiner form (e.g. YOUR, THEIR)
DT0	general determiner (e.g. THESE, SOME)
DTQ	wh-determiner (e.g. WHOSE, WHICH)
EX0	existential THERE
ITJ	interjection or other isolate (e.g. OH, YES, MHM)
NN0	noun (neutral for number) (e.g. AIRCRAFT, DATA)
NN1	singular noun (e.g. PENCIL, GOOSE)
NN2	plural noun (e.g. PENCILS, GEESE)
NP0	proper noun (e.g. LONDON, MICHAEL, MARS)
NULL	the null tag (for items not to be tagged)
ORD	ordinal (e.g. SIXTH, 77TH, LAST)
PNI	indefinite pronoun (e.g. NONE, EVERYTHING)
PNP	personal pronoun (e.g. YOU, THEM, OURS)

PNQ wh-pronoun (e.g. WHO, WHOEVER)
 PNX reflexive pronoun (e.g. ITSELF, OURSELVES)
 POS the possessive (or genitive morpheme) 'S or '
 PRF the preposition OF
 PRP preposition (except for OF) (e.g. FOR, ABOVE, TO)
 PUL punctuation - left bracket (i.e. (or [)
 PUN punctuation - general mark (i.e. . ! , ; - ? ...)
 PUQ punctuation - quotation mark (i.e. ` ' ")
 PUR punctuation - right bracket (i.e.) or])
 TOO infinitive marker TO
 UNC "unclassified" items which are not words of the English lexicon
 VBB the "base forms" of the verb "BE" (except the infinitive), i.e. AM, ARE
 VBD past form of the verb "BE", i.e. WAS, WERE
 VBG -ing form of the verb "BE", i.e. BEING
 VBI infinitive of the verb "BE"
 VBN past participle of the verb "BE", i.e. BEEN
 VBZ -s form of the verb "BE", i.e. IS, 'S
 VDB base form of the verb "DO" (except the infinitive), i.e.
 VDD past form of the verb "DO", i.e. DID
 VDG -ing form of the verb "DO", i.e. DOING
 VDI infinitive of the verb "DO"
 VDN past participle of the verb "DO", i.e. DONE
 VDZ -s form of the verb "DO", i.e. DOES
 VHB base form of the verb "HAVE" (except the infinitive), i.e. HAVE
 VHD past tense form of the verb "HAVE", i.e. HAD, 'D
 VHG -ing form of the verb "HAVE", i.e. HAVING
 VHI infinitive of the verb "HAVE"
 VHN past participle of the verb "HAVE", i.e. HAD
 VHZ -s form of the verb "HAVE", i.e. HAS, 'S
 VM0 modal auxiliary verb (e.g. CAN, COULD, WILL, 'LL)
 VVB base form of lexical verb (except the infinitive)(e.g. TAKE, LIVE)
 VVD past tense form of lexical verb (e.g. TOOK, LIVED)
 VVG -ing form of lexical verb (e.g. TAKING, LIVING)
 VVI infinitive of lexical verb
 VVN past participle form of lex. verb (e.g. TAKEN, LIVED)
 VVZ -s form of lexical verb (e.g. TAKES, LIVES)
 XX0 the negative NOT or N'T
 ZZ0 alphabetical symbol (e.g. A, B, c, d)

APPENDIX A

UCREL CLAWS C7 Tagset

APPGE	possessive pronoun, pre-nominal (e.g. my, your, our)
AT	article (e.g. the, no)
AT1	singular article (e.g. a, an, every)
BCL	before-clause marker (e.g. in order (that), in order (to))
CC	coordinating conjunction (e.g. and, or)
CCB	adversative coordinating conjunction (but)
CS	subordinating conjunction (e.g. if, because, unless, so, for)
CSA	as (as conjunction)
CSN	than (as conjunction)
CST	that (as conjunction)
CSW	whether (as conjunction)
DA	after-determiner or post-determiner capable of pronominal function (e.g. such, former, same)
DA1	singular after-determiner (e.g. little, much)
DA2	plural after-determiner (e.g. few, several, many)
DAR	comparative after-determiner (e.g. more, less, fewer)
DAT	superlative after-determiner (e.g. most, least, fewest)
DB	before determiner or pre-determiner capable of pronominal function (all, half)
DB2	plural before-determiner (both)
DD	determiner (capable of pronominal function) (e.g. any, some)
DD1	singular determiner (e.g. this, that, another)
DD2	plural determiner (these, those)
DDQ	wh-determiner (which, what)
DDQGE	wh-determiner, genitive (whose)
DDQV	wh-ever determiner, (whichever, whatever)
EX	existential there
FO	formula
FU	unclassified word
FW	foreign word
GE	germanic genitive marker - (' or's)
IF	for (as preposition)
II	general preposition

IO	of (as preposition)
IW	with, without (as prepositions)
JJ	general adjective
JJR	general comparative adjective (e.g. older, better, stronger)
JJT	general superlative adjective (e.g. oldest, best, strongest)
JK	catenative adjective (able in be able to, willing in be willing to)
MC	cardinal number, neutral for number (two, three..)
MC1	singular cardinal number (one)
MC2	plural cardinal number (e.g. sixes, sevens)
MCGE	genitive cardinal number, neutral for number (two's, 100's)
MCMC	hyphenated number (40-50, 1770-1827)
MD	ordinal number (e.g. first, second, next, last)
MF	fraction, neutral for number (e.g. quarters, two-thirds)
ND1	singular noun of direction (e.g. north, southeast)
NN	common noun, neutral for number (e.g. sheep, cod, headquarters)
NN1	singular common noun (e.g. book, girl)
NN2	plural common noun (e.g. books, girls)
NNA	following noun of title (e.g. M.A.)
NNB	preceding noun of title (e.g. Mr., Prof.)
NNL1	singular locative noun (e.g. Island, Street)
NNL2	plural locative noun (e.g. Islands, Streets)
NNO	numeral noun, neutral for number (e.g. dozen, hundred)
NNO2	numeral noun, plural (e.g. hundreds, thousands)
NNT1	temporal noun, singular (e.g. day, week, year)
NNT2	temporal noun, plural (e.g. days, weeks, years)
NUU	unit of measurement, neutral for number (e.g. in, cc)
NUU1	singular unit of measurement (e.g. inch, centimetre)
NUU2	plural unit of measurement (e.g. ins., feet)
NP	proper noun, neutral for number (e.g. IBM, Andes)
NP1	singular proper noun (e.g. London, Jane, Frederick)
NP2	plural proper noun (e.g. Browns, Reagans, Koreas)
NPD1	singular weekday noun (e.g. Sunday)
NPD2	plural weekday noun (e.g. Sundays)
NPM1	singular month noun (e.g. October)
NPM2	plural month noun (e.g. Octobers)
PN	indefinite pronoun, neutral for number (none)
PN1	indefinite pronoun, singular (e.g. anyone, everything, nobody, one)
PNQO	objective wh-pronoun (whom)

PNQS	subjective wh-pronoun (who)
PNQV	wh-ever pronoun (whoever)
PNX1	reflexive indefinite pronoun (oneself)
PPGE	nominal possessive personal pronoun (e.g. mine, yours)
PPH1	3rd person sing. neuter personal pronoun (it)
PPHO1	3rd person sing. objective personal pronoun (him, her)
PPHO2	3rd person plural objective personal pronoun (them)
PPHS1	3rd person sing. subjective personal pronoun (he, she)
PPHS2	3rd person plural subjective personal pronoun (they)
PPIO1	1st person sing. objective personal pronoun (me)
PPIO2	1st person plural objective personal pronoun (us)
PPIS1	1st person sing. subjective personal pronoun (I)
PPIS2	1st person plural subjective personal pronoun (we)
PPX1	singular reflexive personal pronoun (e.g. yourself, itself)
PPX2	plural reflexive personal pronoun (e.g. yourselves, themselves)
PPY	2nd person personal pronoun (you)
RA	adverb, after nominal head (e.g. else, galore)
REX	adverb introducing appositional constructions (namely, e.g.)
RG	degree adverb (very, so, too)
RGQ	wh- degree adverb (how)
RGQV	wh-ever degree adverb (however)
RGR	comparative degree adverb (more, less)
RGT	superlative degree adverb (most, least)
RL	locative adverb (e.g. alongside, forward)
RP	prep. adverb, particle (e.g. about, in)
RPK	prep. adv., catenative (about in be about to)
RR	general adverb
RRQ	wh- general adverb (where, when, why, how)
RRQV	wh-ever general adverb (wherever, whenever)
RRR	comparative general adverb (e.g. better, longer)
RRT	superlative general adverb (e.g. best, longest)
RT	quasi-nominal adverb of time (e.g. now, tomorrow)
TO	infinitive marker (to)
UH	interjection (e.g. oh, yes, um)
VB0	be, base form (finite i.e. imperative, subjunctive)
VBDR	were
VBDZ	was
VBG	being

VBI	be, infinitive (To be or not... It will be ..)
VBM	am
VCN	been
VBR	are
VBZ	is
VD0	do, base form (finite)
VDD	did
VDG	doing
VDI	do, infinitive (I may do... To do...)
VDN	done
VDZ	does
VH0	have, base form (finite)
VHD	had (past tense)
VHG	having
VHI	have, infinitive
VHN	had (past participle)
VHZ	has
VM	modal auxiliary (can, will, would, etc.)
VMK	modal catenative (ought, used)
VV0	base form of lexical verb (e.g. give, work)
VVD	past tense of lexical verb (e.g. gave, worked)
VVG	-ing participle of lexical verb (e.g. giving, working)
VVGK	-ing participle catenative (going in be going to)
VVI	infinitive (e.g. to give... It will work...)
VVN	past participle of lexical verb (e.g. given, worked)
VVNK	past participle catenative (e.g. bound in be bound to)
VVZ	-s form of lexical verb (e.g. gives, works)
XX	not, n't
ZZ1	singular letter of the alphabet (e.g. A,b)
ZZ2	plural letter of the alphabet (e.g. A's, b's)