A Study of Parsing Process on Natural Language Processing in Bahasa Indonesia

Elisa Margareth Sibarani
Informatics Engineering Study Program
Del Institute of Technology
Toba Samosir Regency, North Sumatra, Indonesia
elisa@del.ac.id, elisamargareth.sibarani@gmail.com

Mhd. Nadial[1], Evy Panggabean[2], Meryana S[3]
Informatics Engineering Study Program
Del Institute of Technology
Toba Samosir Regency, North Sumatra, Indonesia
{[1]if10007, [2]if10010, [3]if10073}@students.del.ac.id

***Abstract*—Research on Natural Language Processing (NLP) in Indonesian is still limited and the results of available research that can be used for further research are also limited. In a series of natural language processing, the initial step is parsing the sentence in a particular language based on the grammar in order to help understanding the meaning of a sentence. This research aims to produce a simulation of Indonesian parser by adapting the process which was conducted by using Collins Algorithm. The three main stages are: 1) preprocessing to generate corpus and events files, 2) lexical analysis to convert the corpus into tokens, and 3) syntax analysis to build parse tree that requires file events to calculate the probability of the grammar by count the occurrence frequency on file events to determine the best sentence trees. An evaluation was performed to the parser using 30 simple sentences and the outcomes were able to generate a corpus file, file events, parse-tree and probability calculations. Nevertheless some sentences could not be parsed completely true because of the limitations of the Treebank file in Indonesian. Some future works are to develop complete and valid Treebank and Lexicon files.**

Keywords-Natural Language Processing (NLP), Parser, Preprocessing, Lexical Analysis, Syntax Analysis

I. Introduction

Natural Language Processing (NLP) is the area of interdisciplinary research that aims to develop a computer program that can generate text in a natural language and speech [3]. The goal is to enable computers to communicate with humans in the same way humans communicate with other humans [5]. NLP perform natural language processing with multiple stages and implemented into an application. Several stages include Parser, Part Of Speech Tagger and Named Entity Recognizer [6]. Conducted research related to Natural Language Processing includes computational linguistics to the development of applications that can process human language such as sentence understanding, machine translation, probabilistic parsing and tagging, biomedical information extraction, grammar induction, word sense disambiguation, automatic question answering, text and speech generation , information retrieval and text clustering [3]-[6]. Obtained from several sources, the existing research in NLP has focused on the development of processing for English, Arabic, Chinese and German [7].

Development for Indonesian research broadly described as follows:

1) Build a machine Grammar Induction to learn grammar from corpora where these machines are built using genetic algorithm [1].
2) Registering Indonesian study that examines the related computational linguistics in Indonesian [2]. The study covers four major things, which are: a) generating corpus contains 2,200,818 words formed from 74,559 unique words, b) morphological structural analysis of words in Indonesian and stemming algorithms based on it, c) MMTS (Multilingual Machine Translation System) related to the Indonesian Electronic Dictionary (KEBI) consisting of 22,500 words and 43,500 derivative words, d) Indonesian Grammar: syntax analyzer to construct sentences in Indonesian.
3) Using Collins Parser to parse and part-of-speech tagging for Indonesian. The resulting Parser and POS Tagger can parse all sentences that fit with the structure of the sentence pattern SPOK (Subject, Predicate, Object, and Description) and are also able to parse several sentences that have unique structures, covering command sentences and complex sentences [5].

Parser is important in the series of NLP that is useful for understanding the meaning of human language by determining the grammar of a language. It will form a grammar parse tree so we are able to conclude the main idea of the sentence [4]. Thus, a research on Collins parser algorithm needs to be done by adapting it to parse sentences in Indonesian. This paper will describe the process of developing parser for Indonesian by using Collins Algorithm. We use the model 1 from Collins Algorithm and adapt it based on research presented in [5]. The differences from previous research are: We develop the parser from the scratch because Indonesian parser presented in [5] is still not freely available. Also we use lexicon file from 'Kamus Besar Bahasa Indonesia' (KBBI) to be used by the algorithm and with the simplification made to the probability formula referred directly from dissertation described in [11]. The purpose of this research is to produce a simple simulation of Indonesian parser by adapting the process required by Collins Algorithm. Thus, a study needs to be done in order to implement those algorithms.

In Chapter 1 we present the background of this research, continue with chapter 2 in which we elicit knowledge and complete information about Collins Algorithm and all components needed for the algorithm. Based on the study result, we discuss and analyze our own approach to do the parsing process and introduce it in Chapter 3. Thereby, we describe clearly how the parsing for Indonesian can be done. Chapter 4 describes the result of the simulation process and explains some feedback which serves as an evaluation of the proposed method. Chapter 5 summarizes our study and describes future directions that we proposed for future research.

II. LITERATURE STUDY

A. *Natural Language Processing (NLP)*

To support NLP, it requires multiple components developed for processes human language to make it more effective. The available tool that integrates natural language processing component is CoreNLP Stanford. The following are the components that build NLP [6]-[7], namely:

1) Parser, is a tool that is used to parse words in a sentence and gives the tag based on the rules of the grammar.
2) Part-Of-Speech Tagger (POS Tagger) is NLP tools to tag every word that has been parsed in a sentence based on Brown tag set or Penn Treebank tag set in order to obtain the relationship between words. The tags that are given are more complex and detailed than the tag by the parser, so that POS Tagger can distinguish variations in verb forms, variations of singular and plural nouns, and conjunctions variations.
3) Named Entity Recognizer (NER), is a tool that is used to identify a noun word and able to acknowledge whether it is a name of a person, organization or a country.
4) Coreference Resolution System, is a tool used to determine the similarity of any word to recognize the terms used in the sentence.
5) Semantic Analyzer, is a tool that can be used to identify the linguistic meaning of words and extract information / facts from a set of words.

B. *Parser*

NLP parser is a program that develops the grammatical structure of the sentence, for example, a group of words which is a series of words (phrases) and which word is the subject or objects of a verb and generates the grammar tree of it [7]. While doing parsing process, it uses the grammar rules consists of several specific types [4]: a) Context-Free Grammar (CFG), which has four parameters (4-tuple) that is a collection of non-terminal symbols, terminal symbols, production rules, and initial symbols, b) Lexicalized Context-Free Grammar (LCFG), CFG developed into LCFG which can represent a pattern tree grammar. LCFG has five parameters (5-tuple) that are non-terminal symbols, terminal symbols, initial symbol, production rules and the tree grammar, c) Lexicalized Stochastic Context-Free Grammar (SLCFG), has eleven parameters (11-tuple). Six parameters are the additional to the LCFG parameters that are the probability of increase or change possibility that may occur in the tree on production rules, d) Probabilistic Context-Free Grammar (PCFG), has five parameters (5-tuple) is a collection of non-terminal symbols, terminal symbols, production rules, initial symbol, and all possible probabilistic value for each production rules.

List of criteria to define a good parser should meet the following: 1) able to identify ambiguity, 2) using grammar or treebank file to support the process, 3) efficient to do the process, 4) the results can be traced. To perform parsing process, it takes several components, namely [4]: 1) Treebank file, which contains a small treebank obtained by manual analysis of the sentence. This file will be processed into events file, 2) Events files, generated by preprocessing and then will be used to determine the probability of each parse tree, 3) Grammar file contains a set of production rules for a specific language, 4) Non-terminal symbols file, to store a list of non-terminal symbols which are used to tag the word, 5) Corpus files, contains words with the tag which is obtained by preprocessing the sentence and then provide the appropriate tag to the word in the lexicon, 6) Lexicon files, is a simple dictionary, a glossary of terms in a specific domain, will be used in the parsing process which serves as a dictionary that would be used to determine the tag of each word in the sentence [5]. Lexicon files used in this research based on KBBI consist of 1,000 words. In addition, Indonesian has words that can change depending on the affix given to the root word, and it would require some file to identify the tag of those words [4]: a) prefix file, contains a set of prefixes, b) suffix file, contains a set of suffixes, c) confix file contains a set of prefix-suffix pair. Meanwhile the lexicon file only contains the root word, therefore we need prefix, suffix, and confix file to identify the root word of the word contains affixes.

C. *Parsing Method*

Several methods for parsing process explained as follows [4], [9], and [10]:

1) Top-Down Parsing was done by parsing a sentence starts from complete component to the smallest component [9]. This is done continuously until we get the smallest component of the sentence that is the word. Steps to perform the decomposition of the top-down methods in Fig. 1 are: 1) start with the start symbol, 2) conducting decomposition with grammar rules to obtain the sentential form of the grammar, 3) until we get the terminal symbol of the parse-tree.

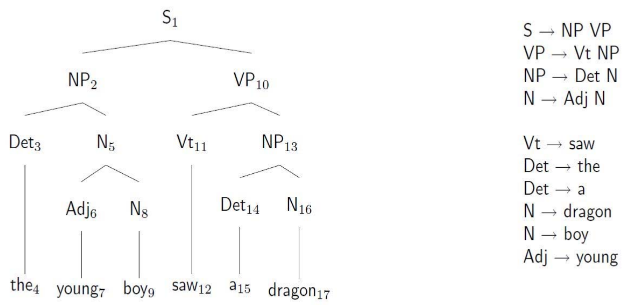


Fig. 1 Top-down Parsing

2) Bottom-Up Parsing is done by taking one word at a time from a given sentence, to be assembled into larger components and done continuously until it forms sentences [9]. Bottom-up process began by backward process with the sentence being parsed using the

grammar rules until reaches the start symbol. Thus the bottom-up method works the other way round from top-down. Steps to perform decomposition with bottom-up method in Fig. 2 are: 1) start with the sentence as the initial sentential form, 2) wait until a sentential form, start symbol can do several things, which are: a) reviewing through the input, until we recognize something that relates to the right hand side (RHS) of the production rules, b) using production rules in reverse, override RHS sentential form that appears in the rule lhs (known as Reduction).

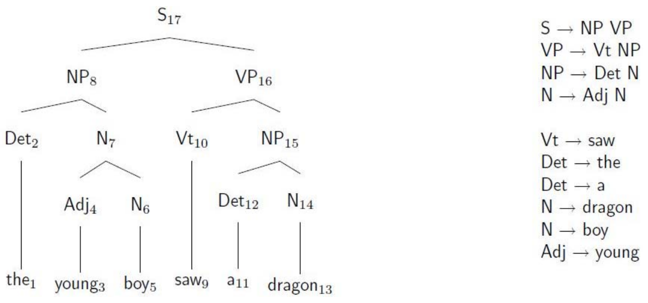


Fig. 2 Bottom-up Parsing

Numbering on the parse tree starts from bottom to the top and so on. The weakness of this method is unable to handle the empty production (eg. non-terminal → 0), so we cannot analyze the production rules of the sentence.

D. Collins Algorithm

Research on the parsing/decomposition process by Collins in [4]-[10] were based on bigram dependency probability that is two strings in the token. Bigram probability calculation depends on two related words. Bigram calculations in this study were calculated based on tags between two words that are related each other (adjacent). Bigram calculation results will be used to calculate the probability of the parse tree. To use Collins Algorithm, it is necessary to use the algorithm Collins PCFG grammar rules so that it can be shown how to extend the grammar into lexicalized grammar. Collins algorithms use history-based approach, which cares for some things which are: a) focus on the head, b) top-down derivation, c) use the conditional probability, in which event A occurs after event B will occur is denoted by:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}, P(B) > 0 \tag{1}$$

This condition is used to determine the chances of the same grammar emerged in the sentence. For example, a grammar as follows:

S → VB NN
S → VB JJ
S → VB

The probabilities result will be:

S → VB NN	0.6
S → VB JJ	0.3
S → VB PU	0.1

By using this formula, we can obtain the grammar occurrence probability from entire grammar that we had from the learning data, with each grammar occurrence probability derived from S = 1.0. Parser Collins was based on PCFG in which it is essentially based on statistical parsing model that defines the conditional probability:

$$P(T \mid S) \tag{2}$$

The purpose of this probabilistic calculation is to obtain the best of each candidate tree T for a sentence S, it can be stated that:

$$T_{best} = \arg\max_{T} P(T \mid S)$$

$$T_{best} = \arg\max_{T} \frac{P(T,S)}{P(S)} \tag{3}$$

$$T_{best} = \arg\max_{T} P(T,S)$$

The formula needed to obtain the probability tree of the sentence (P (T, S)):

$$\mathrm{P(T,S)} = \prod_{i=1..n} P(RHS_i \mid LHS_i) \tag{4}$$

The formula needed to obtain a probability tree on the right with the left (P (RHS | LHS)):

$$\mathrm{P(RHS \mid LHS)} = \frac{Count(LHS \rightarrow RHS)}{Count(LHS)} \tag{5}$$

E. Related Work

In [5], it was a research on the parsing process using Collins Algorithm in which it states that the algorithm can be used for Indonesian, but need to do additional initial processing (preprocessing) to produce the required input files, with description as follows:

1) Generate the Treebank file manually due to the lack Treebank for Indonesian. The Treebank file was made the same with Treebank format used by Collins Parser.
2) Generate Events file from Treebank file heuristically which stores possibility of element dependencies in a sentence. File events are used to calculate the probability of the emergence of grammar to know the probability of the tree of a sentence.
3) Build a Grammar file which is generated through the Treebank coupled with the Grammar used by Collins Parser though in English but still logically be used for Indonesian.
4) Build a Non-terminal symbol file which was taken from non-terminal symbols used by Collins Parser and was adapted to the Indonesian.
5) Build a Corpus file contains sentences along with the tag of each word; the number of words in a sentence which is obtained from a text file that contains a collection of sentences in Indonesian.
6) Create a Lexicon file contains root word along with the tag.

III. Analysis

To parse sentence in Indonesian, we need initial process called preprocessing that aims to generate input files needed for parsing. To generate the parse tree, the parsing process can be seen in Fig. 3.

Indonesian parser built in [5] using Collins Algorithm with some adjustments was made to the input document in the preprocessing to generate corpus and events file. In addition to those documents, there is a Grammar and Treebank files which were adapted from Collins and translated manually. Parser simulator in this study in principle: a) uses the method from Collins Algorithm with Model 1 and uses preprocessing, b) Lexicon file are obtained from KBBI, c) the probability formulas that are used in this research were simplification based on [11]. While in [5] using all formulas on the Collins Algorithm to parse, due to the lack of a detailed description of how to adapt the formula.

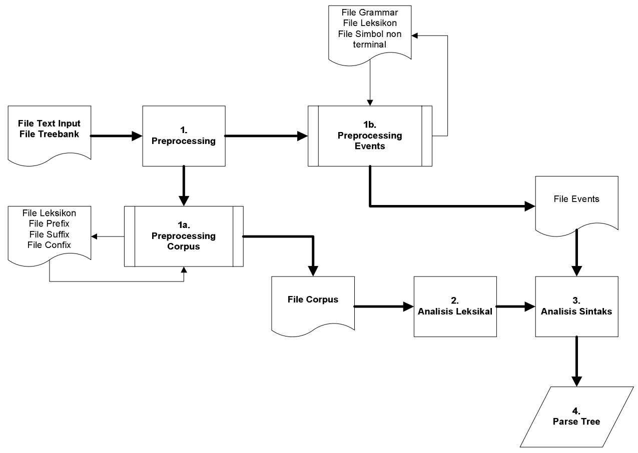


Fig. 3 Parsing process analysis

A. *Input Analysis*

Some of the files needed by the parser that is stored in a text file with extension. txt described as follows:

1) Grammar file contains grammar of a particular language. It is used to determine the frequency of appearance of specific grammar in Treebank and determine the true grammar in the Treebank. Grammar file contents can be seen in Fig. 4. Grammar file that is used in this study based on grammar in [5]. Grammar rules consist of three elements which are non-terminal symbols generated by the head of the sentence. For example, the following sentence: ‘Saya makan nasi’. The main verb of the sentence is the word 'makan' that would be considered a Sentence (S). Symbol S is the start symbol of the sentence and can be reduced to sub-level form of terminal symbols and non-terminal.

```
L ADJP ADJP CC
L ADJP ADJP JJ
L ADJP ADJP NP
L ADJP ADVP IN
L ADJP ADVP JJ
L ADJP ADVP RB
L ADJP ADVP VB
```

Fig. 4 List of Grammar

S has three sub-levels which form a grammar. Grammar patterns that can be generated from the sentence can be seen in Fig. 5.

“Saya makan nasi”
Saya -> NN
Makan -> VB
Nasi -> NN
The grammar result:
L S NN VB
R S VB NN
In which,
S is the sentence symbol as initial symbol
L is left side rule
R is right side rule
followed by non-terminal symbol NN, VB and NN.

Fig. 5 Grammar File

2) Lexicon file is created using word dictionary vocabulary KBBI by: a) delete the phrase in the dictionary (if any), b) sorting the corresponding alphabet, c) changing the format of the word: [word] [word_tag] [classification], d) classification of word tag will be determined by probabilistic classification. Lexicon file will be used in preprocessing stage to build corpus and specify a valid word in a sentence. Lexicon file is used to determine the tag of sentences before stored it in corpus. If we can found specific word in lexicon, the tag for that word will be based on the tag in lexicon. If the word is not found, then additional process to check the affixes on words must be done, or give tag NN (noun) and add it to the lexicon. The probability of each word is calculated from the frequency of words that appear in each corpus. Example of lexicon file that will be used in the parsing process is shown in Fig. 6.

```
aba-aba NN 0
aba NN 0
abad NN 0
abadi ADJ 0
abai ADJ 0
abang NN 0
abdi NN 0
```

Fig. 6 Lexicon File

3) Non-terminal symbol file stores the non-terminal symbol used as a tag of a series words in order to know the grammar of a sentence and it can decomposed into terminal symbols which are considered as tokens. Token is a word that has a terminal symbol and do not

classified as a phrase. Non-terminal symbols in TABLE 1 along with a lexicon will define the validation of words to build a corpus in preprocessing.

TABLE 1. NON-TERMINAL SYMBOL

Symbol	Description
S	*Sentence*
ADJP	Adjective Phrase
ADVP	*Adverb Phrase*
NP	*Noun Phrase*
SBAR	*Relative Clause*
SBARQ	*Relative Clause after question word*
VP	*Verb Phrase*

Some terminal symbols that will be used in the parsing process [4]-[8] can be seen in TABLE 2.

TABLE 2. TERMINAL SYMBOL

Symbol	Word Tag	Description
JJ	Adjektiva	Kata sifat e.g. baru, unik
RB	Adverbia	Kata keterangan e.g. sekarang
AR	Artikula	Kata sandang e.g. si, sang
CC	Konjungtor Koordinatif	Kata hubung yang menghubungkan klausa pada kalimat majemuk setara e.g. dan, lalu
CS	Konjungtor Subordinatif	Kata hubung pada kalimat majemuk bertingkat e.g. ketika, walaupun
MD	Modal	Kata keterangan modalitas e.g. boleh
PR	Pronomina	Kata ganti e.g. saya
WH	Kata Tanya	Kata yang digunakan untuk menanyakan sesuatu e.g. siapakah
NN	Nomina	Kata benda e.g. pensil, meja
CD	Numeralia	Kata bilangan e.g. seribu
IN	Preposisi	Kata depan e.g. di, ke, dari
UH	Interjeksi	Kata seru Contoh : ah
RP	Partikel	Kata tugas partikel e.g. pun, per
VB	Verba	Kata kerja e.g. berlari, makan
AUX	Kata bantu	Kata bantu e.g. akan, dapat
FW	Kata asing	Kata asing e.g. download, notebook
PU	Tanda baca	Tanda baca e.g. .(titik), ,(koma), '(kutip)
SYM	Simbol matematika	+, #, $
X	Unknown	Kata yang tidak dapat diprediksi jenis katanya
NNC	Countable common nouns	Kata untuk benda yang masih dapat dihitung e.g. buku, rumah, karyawan

4) Corpus file contains a set of words that already have a tag. Corpus file has a pattern:

[number of word in sentence] [word] [word_tag]

Corpus file will be processed into a parse tree using a parser through lexical analysis and syntax analysis. Corpus file is generated from a text file through the preprocessing. Example of contents in corpus file can be seen in Fig. 7.

```
7 Dia PR mendapatkan VB ranking NN satu
CD se AR kabupaten NN . PU
```

Fig. 7 Corpus File

5) Treebank file is a document that contains a collection of parse trees of sentences that will be used to enrich the lexicon and grammar. Grammar file is obtained by performing parse to a simple sentence manually. Treebank that is used in this study based on Treebank in [5] contains ± 42 Treebank and all files are stored in a text file. An example of its content can be seen in Fig. 8. Treebank files is processed in preprocessing events and used to obtain the frequency of a grammar rules.

```
(S(NP(PR Kamu))(ADVP(JJ tentu)(RB
sering))(PU ,)(ADVP(RB bahkan)(JJ
mungkin)(AR setiap)(NN hari))(PU ,)(>VB
mendengarkan) (NN berita)(ADVP(IN di)(NN
televisi)(>CC atau)(IN di)(NN radio))(PU .))
```

Fig. 8 Treebank File

Legends in Fig. 8 are:

a) '>' is used as a mark for the head of each parse level (core meaning).
b) '()' is a bracket used as a mark containing the head level of word, word tag and word.
c) S is used as initial symbol.

6) Events file is the tree representation of a sentence that is used to calculate the probability of grammar rules that exist in the grammar file [4]. The probability value of the grammar rules will be used to calculate the probability of occurrence a specific parse tree of a sentence. Description of the code numbers used on file events: a) code digit 6 indicates the reading sentences that made the sentence tree, b) code digit 3 indicating unary event code, which means that the tree has only one child or the relationship between the node with his head. Event unary obtained from:

3 [headword] [headtag] [label] [headchild →label] 00000 00000

c) code digit 2 indicating event dependency for three non-terminal symbols (grammar). Event dependency is obtained from:

2 [word_node_child] [tag_node_child] [headword] [headtag] [label_node_child] [label] [headchild →label] 000000 [direction_grammar] 0 0

Event unary and event dependency will generate grammar patterns. For example, unary event:

3 menanam VB S VB 00000 00000

The grammar's result,

S → VB

Then, the event dependency:

2 Dia PR menanam VB NP S VB 000000 110 0 0

The grammar's result,

S $\rightarrow$ NP VB

Thus the process is continually done for every event unary and event dependency. Then the resulting grammar will be calculated and used as a probability value to each grammar pattern.

7) Affix file, is a file that contains possible affix in Indonesian that is used in morphological analysis in the parsing process [4]. Morphological analysis aims to separate the root words and affixes contained in the word.

B. Process Analysis

The parsing process for Indonesian is explained as follows:

1) Preprocessing is performed to generate the input files needed to do the parsing. Preprocessing consists of two phases:
   a) Generating corpus files: 1) identify the sentences by read each word from the text file and are saved into the corpus file (.cps), 2) if it is punctuation (such as dot (.), Question words (?) and exclamation (!)) then it will be identified as the end of the sentence, 3) create a corpus file by determining the number of words of each sentence and provide tags for each word by check it with the lexicon file.
   b) Generating events file: 1) classify sentences based on sentence trees by searching the symbol '>' and the tag PU as a mark for end of a sentence, 2) create the events file by determining the event unary and event dependency of each sentence, 3) check the grammar and lexicon of the Treebank file and compares to the grammar and lexicon that already exist. If the grammar in the Treebank file is not listed in grammar file then it will be added to enrich the grammar file. As well as the checking of the existing tags on Treebank file. If the existing tags on Treebank files are not listed in the lexicon, then the tag and the word will be added to the lexicon to enrich the lexicon file.
2) Lexical Analysis, to produce a token that is used as an input to syntax analysis. Lexical analysis needs corpus file that will be used as input, with the steps described below:
   a) Corpus file will be read as many as the number of words in a sentence. At the beginning of every sentence in the corpus, there are tags that indicate the number of word pairs and tags that should be read. The format for corpus file is written below:
   [number_word_in_sentence] [word] [word_tag]
   Suppose [number_word_in_sentence] = n, and [word_tag] = tag, then the number of pairs of words and tags is the value of n.
   b) Check the tag words that exist in the corpus file is already same with the tag word used in the non-terminal symbol file. Each tag will be compared with the list of non-terminal symbol, with condition: 1) if the tag is registered then words and tags will be made into a token, 2) if the tag in the corpus has not been registered yet on the non-terminal symbol file then those tag will be added to the non-terminal symbol and then words and tags will be made into a token.
3) Syntax analysis is used to calculate the probability tree of sentence that is possible to be generated. The steps in the syntax analysis are:
   a) Examine the validity of a token which is given from previous step to the lexicon. If the tag of the word on the token is the same with in the lexicon, then it is valid.
   b) If the tag in the token does not match then it will be checked against affixes. If the token contains affixes then we have to change it to the root word based on suffix, prefix, and confix file. If no affix was found, then check whether it uses NN, if yes, then it is valid word, if not, and then gives the tag NN to the word. Once the word is formed it will be followed by checking against the rules of grammar. Similarly, if the tag is valid then it can continue to the next process.
   c) Checking the grammar rules of the token in order to identify for compliance with grammar in grammar file. Initial conditions to do this process are: 1) check if there is a verb and the first will be made as main verb, 2) if a sentence has no verb, the word with tag AUX is the head of the sentence, 3) if a sentence has no verb and Auxiliary (e.g. 'adalah', 'ialah'), then the first noun that was first discovered was the head of the sentence.
   After the beginning of the sentence is determined, then we do the grammar checking. For example sentence below has the head sentence 'menanam' VB:
   4 Saya PR menanam VB bunga NN . PU
   Then the grammar pattern that must be examined in the grammar file is:
   S VB PR
   S VB NN
   S VB PU
   S VB NN
   d) Calculate the grammar probability of each sentence require events file and Treebank. Treebank file is used to calculate the classification value of a word and then update the classification value which is stored in the lexicon file. While the events file is used to calculate the probability of a sentence so as to know how likely probability trees generated from each sentence. Probabilities obtained will be inserted into events file and replace the previous probability value.
   To calculate the probability for grammar occurrence will use Equation (1). The purpose is to overcome the ambiguity in case several tree has been resulted from one sentence by determining the best tree that can be formed by a parser. To find out the best tree

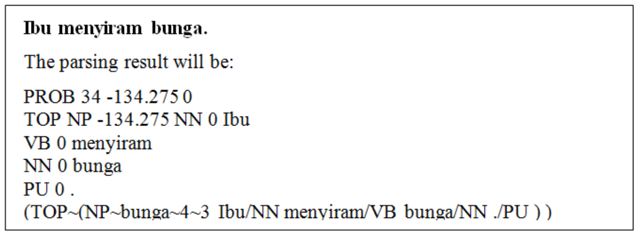
for each sentence, we can use Equation (3), where every opportunity P (T, S) can be obtained from equation (4).

With the probability calculation of grammar based on the sentence, it will obtain the best tree as a result of parsing. The probability calculations in this study aims to determine the best tree for the ambiguous sentence and sentence contains phrases, while simple sentences can be parsed with lexical analysis. This rule is derived from Collins Parser with some adaptive changes according to research [5].

e) Generate the parse tree for sentence according to the result from previous process.

C. *Result Analysis*

The parser is performed to generate a parse tree that will provide information in accordance with a series of natural language processing step. The results of the sentence tree will contain the probability calculation of the tree and the symbol of every word in the sentence. Parse tree that will be generated as the output of the application is shown in Fig. 9.



```
Ibu menyiram bunga.
The parsing result will be:
PROB 34 -134.275 0
TOP NP -134.275 NN 0 Ibu
VB 0 menyiram
NN 0 bunga
PU 0 .
(TOP~(NP~bunga~4~3 Ibu/NN menyiram/VB bunga/NN ./PU ) )
```

Fig. 9 The parsing result

IV. Implementation and Evaluation Result

Results of the analysis described in the previous chapter are implemented as a simulator application that simulates a parser for Indonesian sentence. The parser simulator will process the sentence into a parse tree and will accept input from the user in the form of text. The simulator itself is a desktop application built in Java language. Limitations of the simulator are: 1) receive input file contains Indonesian sentences with the file extension .txt, 2) process simple sentence, 3) able to perform preprocessing to generate corpus and events file and can generate parse-tree based on corpus, 4) conduct simple probability calculations for simple sentence.

The ability of the application was evaluated using 30 sentences that are stored in the file .txt. The sentences consist of: 11 sentences with SPO (Subject-Predicate-Object), 9 sentences with SPK (Subject-Predicate-Complement), 9 sentences with SPOK (Subject-Predicate-Object-Complement), and 1 instruction sentence. Descriptions of the evaluation result performed on these sentences are described as follows:

1) In the corpus preprocessing: a) applications successfully generate corpus file for all sentences correctly. However, tag for some words were undefined because it was not listed in the lexicon file, b) Lexicon files consists of about 1000 words and the update process has not been done automatically.

2) In the parsing process, we conclude that the applications can parse all sentences. The true example of a parse tree is shown below:

```
S
---  NN Ibu
---  >VB memasak
---  NN nasi
---  PU .
```

where, S (sentence) is the initial symbol, '>' is used as a marker of the head level at each level of parsing (the core meaning of the sentence) and NN, PU is a terminal symbol that cannot be parsed again. However, two sentences were parsed incorrectly, which are shown as follows:

a) Mobil baru milik ayah sangat mahal. The parse tree resulted from the sentence is:
(S(NP(NN Mobil)(JJ baru))(>VB milik)(NN ayah)(ADJP(RB sangat)(JJ mahal))(PU .))

The correct parse tree is:
(S(NP(NN Mobil)(JJ baru))(>VB milik)(**ADVP**(NN ayah)(ADJP(RB sangat)(JJ mahal)))(PU .))

b) Angin bertiup dengan kencang. The resulting parse tree for the sentence is:
(S(NN Angin)(>VB bertiup)(ADJP(IN dengan)(JJ kencang))(PU .))

The correct parse tree is:
(S(NN Angin)(**VP**(>VB bertiup))(ADJP(IN dengan)(JJ kencang))(PU .))

The parse tree which is obtained from the process depends on the completeness of Treebank file which impacts the decision to choose the most valid grammar for a sentence. In this study, Treebank files which were used only consist of 42 trees with simple structure that is SPO (Subject, Predicate, and Object). If the available Treebank file is complete which contains complex sentence with SPOK structures (Subject, Predicate, Object, Description), then it is possible to generate correct parse-tree. It is because a parse tree was created based on Treebank which is formed into a tree.

3) In generating events file and probability: a) application is successfully displays detailed information on Treebank files, events file and probability calculations, b) Successfully generate events file, c) applications can generate a probability value of each sentence. However, some calculation are still not as expected because there are sentences that obtain 0 value for the probability of its grammar, d) mistake on the creation of events file and probability values due to the limitations of Indonesian Treebank.

V. Conclusion

Parsing process for Indonesian was done by adapting the Collins Algorithm which was used for English by following a preprocessing to generate corpus and events file. Furthermore, the process is carried out by lexical analysis and syntax analysis and the final results are sentence tree (parse-tree) with the tag of each word in the sentence. The resulting parse-tree was successfully generated for 28 sentences out of 30. It is correct when it used Treebank file to be able to determine properly the appropriate grammar patterns for a specific sentence. Thus the future research is to develop a complete and valid both Indonesian Treebank file and dictionary.

References

[1] Manning, Christopher D., Hinrich Schütze: "Foundations of Statistical Natural Language Processing", The MIT Press, Cambridge, Massachusetts, 1999, ISBN 0 262 13360 1.

[2] Arya Tandy Hermawan, Gunawan, Joan Santoso. "Natural Language Grammar Induction of Indonesian Language Corpora Using Genetic Algorithm", Department of Computer Science Sekolah Tinggi Teknik Surabaya, Surabaya, East Java, 2011.

[3] Mirna Adriani and Ruli Manurung, "A survey of bahasa Indonesia NLP reserach conducted at the University of Indonesia", Faculty of Computer Science University of Indonesia.

[4] Alexander Gelbukh, "Special issue: Natural Language Processing and its Applications", Instituto Politécnico Nacional Centro de Investigación en Computación México, Mexico, 2010.

[5] Rosa Ariani Sukamto, "Penguraian Bahasa Indonesia dengan Menggunakan Pengurai Collins", Master Thesis, Institut Teknologi Bandung, 2009.

[6] http://www.hit.ac.il/staff/leonidm/information-systems/ch68.html diakses pada 20 November 2012.

[7] http://nlp.stanford.edu/software/corenlp.shtml diakses pada 29 September 2012

[8] Mulyono, Anton M.: Tata Bahasa Baku Bahasa Indonesia, Balai Pustaka, Jakarta, 1991.

[9] Setiorini, Retno Asihanti, "Analisis Penggunaan Tata Bahasa Indonesia dalam Penulisan Karya Tulis Ilmiah: Studi Kasus Artikel Ilmiah". Lembagan Ilmu Pengetahuan Indonesia.

[10] Power, James, "Notes on Formal Language Theory and Parsing, "National University of Ireland, Maynooth, Ireland, 2002.

[11] Collins, Michael, "Head-Driven Statistical Models for Natural Language Parsing", MIT Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Association for Computational Linguistics Vol 29 No 4, 2003.