Natural Language Understanding Tools with Low Language Resource in Building Automatic Indonesian Mind Map Generator

Ayu Purwarianti, Athia Saelan, Irfan Afif, Filman Ferdian, and Alfan Farizki Wicaksono

School of Electrical Engineering and Informatics
Bandung Institute of Technology, Jl. Ganeca No. 10, Indonesia

Abstract: Here, we describe our work in developing Indonesian Mind Map Generator that employs several Indonesian natural language understanding tools as its main engine. The Indonesian Mind Map Generator[1]aims to help the user in easily making a Mind Map object. The system consists of several Indonesian natural language understanding tools such as Indonesian POS Tagger, Indonesian Syntactic Parser, and Indonesian Semantic Analyzer. The methods used for developing each of Indonesian natural language understanding tools are devised to such an extend that they are enable to alleviate the low availability of Indonesian language resources. For Indonesian POS Tagger, we employed HMM and subsequently enhanced the result by using affix tree. As for the Indonesian Syntactic Parser, we compared the performance of CYK and Earley parser, which are known as common dynamic algorithms in PCFG. The Indonesian Semantic Analyzer consists of several components such as lexical semantic attachment, reference resolution, and Semantic Analyzer itself that transforms the parse tree result into first order logic representation. In our work, instead of using a rich resource on semantic information for each vocabulary, we defined several rules for the lexical semantic attachment based on POS Tags and certain words. Finally, to develop the Mind Map generator, we used the radial drawing method to visualize the first order logic representation and we also built a Mind Map editor to allow a user in modifying the Mind Map result. To evaluate the result, we conducted the experiments for each component mentioned previously. The POS Tagger accuracy achieved 96.5%, the Syntactic Parser achieved accuracy of 47.22%, and the Semantic Analyzer achieved accuracy of 62.5%. The final result of Mind Map object was evaluated by 5 respondents. The results of evaluationshowed that, for the simple sentence, the Mind Map object can be easily understood.

1. Background

Nowadays, many education systems employ Mind Map symbols in explaining concepts that can be understood easily by the students. The idea of Mind Map is to use picture and color combination, which is compatible with how the brain works[1]. Since Mind Map is a popular concept, people try to develop Mind Map editors to help the other sin drawing a Mind Map. One of the drawbacks is that, in these Mind Map editors, user has to draw the object from scratch, which can demotivate the user to start using the Mind Map editor. To handle such problems, several researches proposed a Mind Map generator tool to help the user in preparing the Mind Map object. By using a Mind Map generator tool, one doesn't have to draw the Mind Map object from scratch. User can edit the result of Mind Map generator tool and shorten the effort to draw the Mind Map object.

Unfortunately, the Mind Map generator tool is only available for English text[2][3]. In English Mind Map generator, the basic approach is to employ natural language understanding tool in transforming English text into other representations such as syntactical representation or semantic representation. There was no research or product on developing Mind Map generator for Indonesian language. In the recent years, there have been several works on developing

[1]The application can be accessed at http://mindmap.kataku.org

Received: June 3rd, 2013. Accepted: September 5th, 2013

the Indonesian natural language understanding tools such as Indonesian POS Tagger, Indonesian Syntactic Parser, and Indonesian semantic analysis. Yet,there is still no research on developing the Mind Map generator. The available research is to employ the first order logic as the result of semantic analysis in the question answering system [4]. Moreover, there is another work that uses the natural language understanding tools in evaluating user input of understanding simple text [5]. In this paper, we describe our approach in developingan Indonesian Mind Map generator using the available POS Tagger, Syntactic Parser, and semantic analysis.

2. Related Works

Below, we describe two related works on the Mind Map generator for English, namely M^{2}Gen[2] and Actor-based Mind-Maps Assembler[3].

A. M^{2}Gen[2]

The concept of M^{2}Gen is to generate the Mind Map object from semantic model taken from a given text. The English text is transformed into a semantic model or meaning representation using several natural language understanding tools such as morphological analysis, parsing and Semantic Analyzer. The complete process is shown in Figure 1. First, the English text is processed by morphological analysis in order to analyze each word into its lemma and affix along with its POS tag. The result of morphological analysis is then processed by parsing

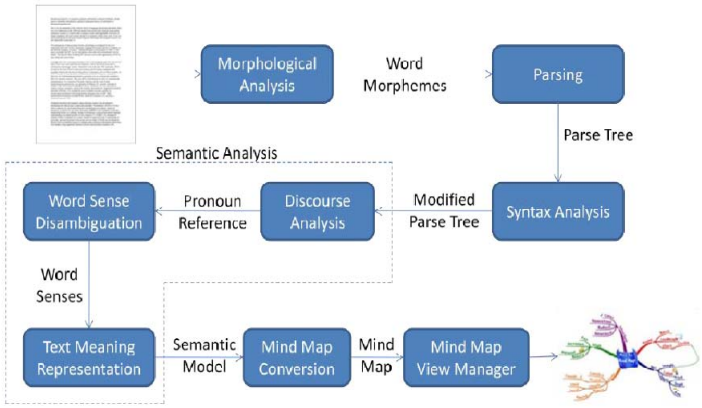

Figure 1. The Architecture of M^{2}Gen [2]

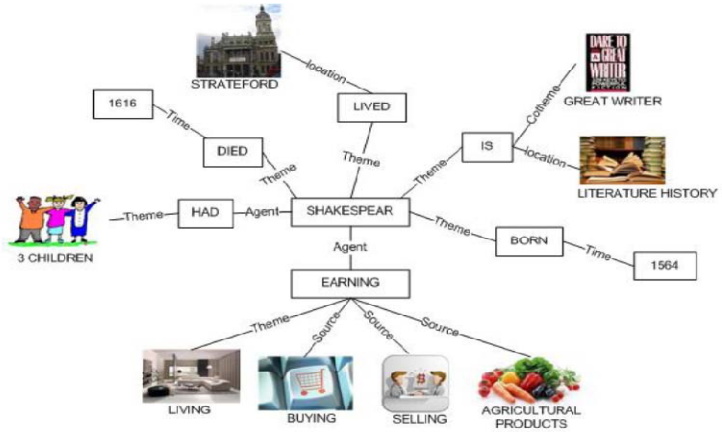

Figure 2. Example of Mind Map Resulted by M^{2}Gen [2]

component by using CFG and top down chart parsing. Since not all parts of parse tree are used, then there is a parse tree modification process in the syntax analysis. The result is then used by semantic analysis to yield the semantic model. The semantic analysis consists of several sub components such as discourse analysis, word sense disambiguation and text meaning representation. The discourse analysis aims to solve the pronoun reference in sentences, the word sense disambiguation aims to select the best sense of a single word, and the text meaning representation aims to transform the parse tree result into the semantic model. The resulted semantic model is then converted into Mind Map figure. The example of generated Mind Map figure is shown in Figure 2.

B. Actor-based Mind-Maps Assembler[3]

Here, the concept of Mind Map is taken from the subject and object of a sentence, which is assumed as the actor. The relation between concepts is taken from the sentence predicate. Thus, the application doesn't need the semantic model of a sentence, it only needs the syntactical parse tree such as shown in Figure below.

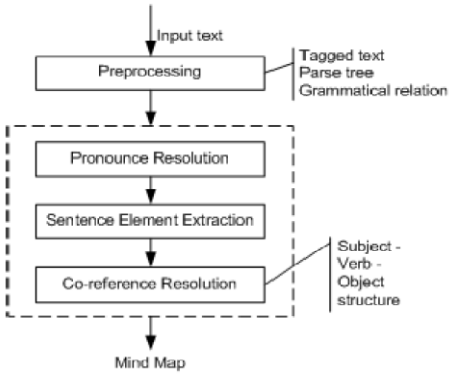

Figure 3. Architecture of Actor-based Mind-Maps Assembler[3]

The result of syntactical parsing which is in the preprocessing component shown in Figure 3, is then processed by three processes in order to select the subject, verb and object structure. The next process is to transform the subject and object into the concept in the Mind Map and the verb into the relation. The Mind Map result example is shown in Figure 4.

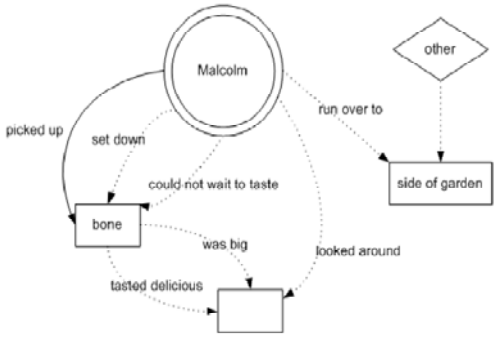

Figure 4. Example of Mind Map as the Result of Actor-based Mind-Maps Assembler[3]

3. Method in Indonesian Mind Map Generator

Here, we described our method in the Indonesian Mind Map generator. The details of several methods are available in other publications[5][6].

We choose to use the meaning representation result as the source for the Mind Map representation such as employed in M^2Gen[2]. The usage of Semantic Analyzer in order to

yield the semantic representation is easier and more accurate than the Syntactic Parser since we already built the Semantic Analyzer. To build the Mind Map object using only Syntactic Parser result will require many defined rules or a tagged corpus which spends more effort than using the semantic representation resulted by the Semantic Analyzer.

The complete architecture of transforming Indonesian text sentence into Mind Map representation is shown in figure below. Here, the whole processes include POS tagger, Syntactic Parser, Semantic Analyzer, reference resolution and Mind Map symbol generator. Each process is processed sequentially. The result of POS tagger is used as the input for syntactic parser, and so on. The technique used in each component is described in following section.

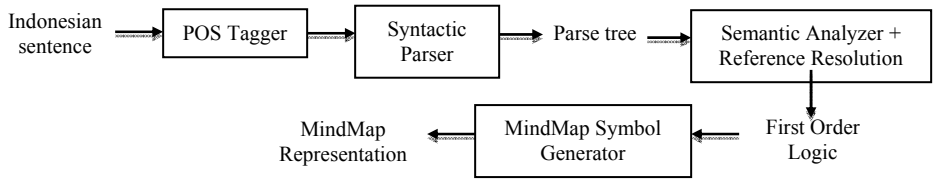

Figure 5. The Architecture of Indonesian Mind Map Generator using Semantic Representation

A. Indonesian POS Tagger

Table 1. POS Tag Set in Indonesian POS Tagger [6]

No	POS	POS Name	Example
1	OP	Open Parenthesis	({[
2	CP	Close Parenthesis	)}]
3	GM	Slash	/
4	;	Semicolon	;
5	:	Colon	:
6	“	Quotation	“ ‘
7	.	Sentence Terminator	. ! ?
8	,	Comma	,
9	-	Dash	-
10	…	Ellipsis	…
11	JJ	Adjective	Kaya, Manis
12	RB	Adverb	Sementara, Nanti
13	NN	Common Noun	Mobil
14	NNP	Proper Noun	Bekasi, Indonesia
15	NNG	Genitive Noun	Bukunya
16	VBI	Intransitive Verb	Pergi
17	VBT	Transitive Verb	Membeli
18	IN	Preposition	Di, ke, dari
19	MD	Modal	Bisa
20	CC	Coor-Conjunction	Dan, atau, tetapi
21	SC	Subor-Conjunction	Jika, ketika
22	DT	Determinier	Para, ini, itu
23	UH	Interjection	Wah, aduh, oi
24	CDO	Ordinal numeral	Ketiga, keempat
25	CDC	Collective numeral	Berlima, berempat
26	CDP	Primary numeral	Satu, sepuluh
27	CDI	Irregular numeral	Beberapa
28	PRP	Personal pronouns	Saya, kamu
29	WP	WH-pronoun	Apa, siapa
30	PRN	Number Pronoun	Kedua-duanya
31	PRL	Locative Pronoun	Sini, sana
32	NEG	Negation	Bukan, tidak
33	SYM	Symbols	&, %, $
34	RP	Particles	Pun, kah
35	FW	Foreign Words	Foreign, computer

POS Tagger is an important tool in understanding an input sentence since the POS tagger result is a foundation for the next step of Syntactic Parser. Without knowing a word POS, the syntactical structure of a sentence can't be defined. The POS tags used in our POS Tagger is modified from the tag set used in [7][8]. The complete POS Tag set used in this research is shown in Table below.

An example of POS tagging result of an input sentence "Kartini lahir di Jepara" is as follow:
Kartini/NNP lahir/VBI di/IN Jepara/NNP

The idea of automatic POS Tagging is how to label POS tag of a word given a list of word as a sentence. Using a manual POS Tagging will spend a lot of resource and might have a risk of labeling inconsistency. Basically, there are two main approaches in an automatic POS Tagging: rule based and statistical based. In a rule based system, the problem here is to define list of POS Tag for words manually without directly considering the real word context. This is a difficult task since it needs several linguistics experts to define the list. While in the statistical based system, the preparation is to make a POS Tag labeled corpus which is easier to build than the list of POS Tag for words such as needed in the rule based system.

Here, we employed HMM (Hidden Markov Model) as the statistical algorithm in our Indonesian POS Tagger. This algorithm is chosen since it is the most employed technique in building POS Tagger of many languages. HMM method was proved to have better running time than any other probabilistic methods [9] in a POS Tagger. The basic idea of HMM method is to select the best list of POS tag for a given sentence. The best list of POS tag means the list with the highest probability score among all the candidate of POS tag list. The probability score is based on the Bayes law which has two models of emission probability and transition probability. The emission probability is the probability of a word given a certain POS Tag. The transition probability is the probability of a POS Tag given a certain previous POS Tag.

P(POS Tag|Words) = P(Words | Pos Tag) P(Pos Tag)…… (eq. 1)

In order to handle the empty probability of P(Words | Pos Tag) or the OOV (Out of vocabulary) problem, we employed a decision tree of affix (suffix and prefix). An example of Prefix tree is shown below. The tree is parsed if the P(Words | Pos Tag) is zero which means that there is no training data for the given word. The tree is parsed from its root. For example, using the prefix tree below, for words "melukis", the tree will be parsed on "m" and "e" and gives result of P(melukis|VBT) = 0.75, P(melukis|VBI) = 0.2 and P(melukis|NN) = 0.05. The probability score for each suffix is calculated from words in the training corpus having a certain suffix such as "me-", "ke-", etc. This technique is adapted from Schmidt[10][11]. The tree used in Schmidt is extended to handle the characteristics of affix in Indonesian language. In Schmidt, the tree only represents the suffix since English doesn't recognize prefix; here, for Indonesian language, we designed the tree to be able to represent prefix and suffix.

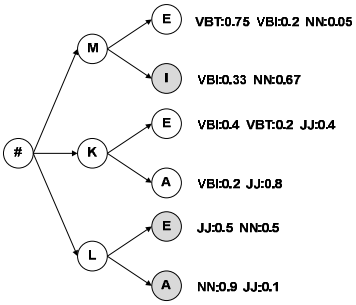

Figure 6. Example of Prefix Tree used in Indonesian POS Tagger with Length 2 [6]

We also enhance the POS Tag prediction for OOV by using the feature of succeeding POS Tag. The succeeding POS Tag feature is used to calculate the emission probability for OOV in the second pass of HMM. Another additional process is to use KBBI Kateglo in filtering the result of prefix tree. The POS Tag candidates resulted by the prefix tree are only used if they are listed in the KBBI Kateglo.

B. Syntactic Parser

A syntactic parser is a component that aims to construct structural relation between words in sentence [12]. Syntactic parser involves two basic things: (1) syntactical grammar; (2) parsing algorithm. Syntactical grammar contains syntactical rules that can be divided into constituent based and dependency based. In our research, we made use the constituent based of probabilistic context free grammar. The constituent type is chosen since it is easier to construct the corpus with constituent information than the dependency one. We employed a PCFG (Probabilistic Context-Free Grammar) to represent Indonesian grammar[12]. The PCFG notation include terminal, non terminal, grammar rules along with its probability score, and the start of non terminal. The terminal represents the POS Tag such as shown in Table 1. The non terminal represents the phrase information such as NP, VP, etc. The grammar rules represent the transition of top non terminal symbol into an array of terminal or non terminal symbols. The example grammar rule is S → NP VP which means that a non terminal S can be parsed into NP dan VP phrases.

As for the parsing algorithm, we compared the probabilistic context-free algorithm of Earley[13][14] and CYK (Cocke-Younger-Kasami) [12][14]algorithms. These two algorithms are chosen since they are common used dynamic programming in the PCFG parser.

The idea of CYK algorithm is to check all possible grammar rules, started from grammar rules for word length of 1, then for word length of 2, and so on, until all words in the sentence are processed. The algorithm is a bottom up parsing and using dynamic programming. Figure below shows the chart illustration of CYK parsing algorithm. First, on the word length of 1, the algorithm will check all possible grammar rules with each single word in the right hand side of the rule. Here, in the example, the resulted ruleisNNP→Kartini.Next, on the word length of 2, the resulted example rules are NP →NNP, VP→VBI and ADVP → IN NNP.

S			
	VP		
NP	VP	ADVP	
NNP	VBI	IN	NP, NNP
Kartini	lahir	di	Jepara

Figure 7. Chart Illustration of CYK Algorithm for "Kartini lahir di Jepara"

The Earley algorithm makes use several operators such as scanner, predictor and completer for each cursor position on every resulted rules. The algorithm is a top down parsing and using dynamic programming. The predictor is activated when the parser finds a non terminal symbol, where the parser will search grammar rules which left hand side is the non terminal symbol. The scanner is activated when the parser finds a terminal symbol. And the completer is activated when all parts of a grammar rule are already parsed.

. Kartini	. lahir	. di	. Jepara	.
S → . NP VP	NP →NNP .	VP →VBI .	ADVP →IN . NP	NP →NNP .
NP → . NNP	S →NP . VP	VP→VP . ADVP	NP → . NNP	ADVP → IN NP .
NP → . NNP	VP → . VP ADVP	ADVP → . IN NP	NP → . NN	VP→VPADVP .
NNP → . Kartini	VP → . VBI	IN → . di	NNP → . Kartini	S → NP VP .
NNP → . Jepara	VBI → . lahir	IN →di .	NNP → . Jepara	
NN → . kota	VBI →lahir .		NNP →Jepara .	
NNP →Kartini .				

Figure 8. Chart Illustration of Earley Algorithm for "Kartini lahir di Jepara"

Both algorithms give result of constituent parse tree such as shown below.

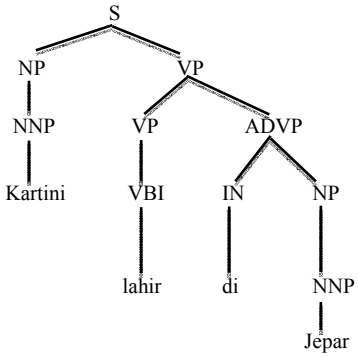

Figure 9. Parse Tree Result of Syntactic Parser for "Kartini lahir di Jepara"

C. Semantic Analyzer and Reference Resolution

Parse tree resulted from the syntactic parser is processed by lexical semantic attachment which gave result of the parse tree along with its lexical semantic information. The advantage of using lexical semantic is to increase the quality of semantic comprehension obtained from text. Next, the parse tree and its lexical semantic information are processed by reference resolution component in order to build relation between sentences. Last component is the Semantic Analyzer itself which target is to transform the parse tree with its lexical semantic information into one knowledge representation, here we used First Order Logic[15][16]. The detail description of each component and the knowledge representation are described in following sections.

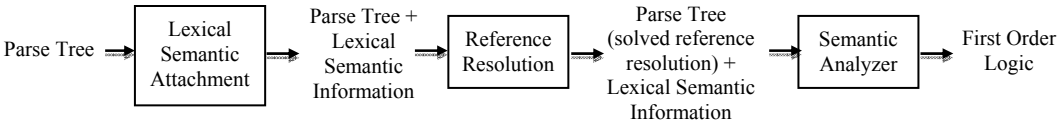

Figure 10. Semantic Analyzer + Reference Resolution Module

1). Knowledge Representation

First order logic is employed as the knowledge representation. The first order logic [15][16]consists of two main terminologies:(1) term that illustrates an object in form of constant or variable; (2) predicate that illustrates preposition. Each object in the first order logic consists of term and predicate. For example, the first order logic of *agent(E,A)* consists of predicate *agent* and terms *E* and *A*. We also employed the flat semantic representation from Hobbs, where the representation is modeled as conjunction of literal[17]. Literal means the predicate that relates term in the first order logic. The advantage of using flat semantic representation is to eliminate the usage of complex logic notation in FOL such as nested quantifiers, disjunction, negation, and so on. Another technique used is the Durme technique for classifying literal into extrinsic and intrinsic literal [18]. The extrinsic literal relates two variables, while the intrinsic literal relates variable and its referent. In Figure 11, *event* is an intrinsic literal, where *E* is the variable and *lahir* is the referent. *Agent (E,A)*is the extrinsic literal where *E* and *A* are the variable.

$$event(E, lahir) \cap agent\ (E, A)$$

Figure 11. Example of Flat Semantic Representation

2). Lexical Semantic Attachment

Lexical semantic is an aspect in Semantic Analyzer that concern on relation among lexeme[12]. It can be used to label a semantic concept to a given word or given phrase in sentence. The semantic concept for a given word can be taken from a thesaurus, for example, the semantic concept for English can be taken from WordNet[2] that can be used to relate each lexeme. Even though, now, there is a research on Indonesian WordNet, but we choose not to use it since the Indonesian WordNet still has lower quality than the English one. Here, we choose to define several rules based on POS tag, certain words or preposition words. Here are some rule examples:

1. Word with POS tag as "NN" (Common Noun) or "NNP" (Proper Noun) are recognized as "object"
2. Certain words such as "tadi", "kemarin", are recognized as "moment"
3. Prepositions such as "di", "ke", and "dari", are recognized as "place"

Several instrinsic literals and extrinsic literals are shown in Table below.

Table 2. List of Instrinsic and Extrinsic Literals used in the Indonesian Mind Map Generator

Intrinsic Literal	**Extrinsic Literal**
λx place (x,y)	λxλy Location (x,y)
	λxλy Source (x,y)
	λxλy Direction (x,y)
λx moment (x,y)	λxλy Time (x,y)
λx person (x,y)	-
λx object (x,y)	-
λx event (x,y)	λxλy Actor (x,y) λxλy Patient (x,y)
λx quantity (x,y)	λxλy Attribute (x,y)
λx property (x,y)	
λx explanation (x,y)	
-	λxλy Manner (x,y)
-	λxλy Complement (x,y)
-	λxλy Comparison (x,y)
-	λxλy Purpose (x,y)

As an example, sentence "Kartini lahir di Jepara" will give lexical attachment results such as:

- Kartini: λa object (a,Kartini)
- lahir: λbλc event(b,lahir)∧ agent(b,c)
- di: λdλe event(d) ∧ place(d,e)
- Jepara: λa object (a,Jepara)

3). Reference Resolution

Reference Resolution is an aspect of discourse processing to build relation between each sentence in text, which concerns in referring reference to its referent[12]. In our Indonesian Mind Map Generator, we only process the pronoun and build the reference resolution to refer pronoun with its referent. We used Hobbs algorithm[17], where it can work by using only syntactic structure of sentence. Since in Indonesian language, a pronoun doesn't concern the referent gender, then it is easier than the English reference resolution. We employed recency in the reference resolution. The recency feature tends to choose the most recent noun entity as the antecedent (referent) of a pronoun. This feature is the easiest and fastest feature to be implemented.

The pronoun processed in our reference resolution includes "dia" (she/he), "ia" (she/he), "beliau" (she/he), "mereka" (they/them) and possessive pronoun recognized by affix "-nya"

[2]http://wordnet.princeton.edu

(her/his). In this reference resolution, the variable pronoun is replaced with its referent and the predicate "object" of referent is replaced by "person". For instance, a paragraph "Kartini lahir di Jepara; Beliau adalah seorang Tokoh Pendidikan" produces this result:

a). Pronoun: λa person (a,Beliau) into λa person (a,Kartini)
b). Referent: λb object (b,Kartini) into λb person (b,Kartini)

4). Semantic Analyzer

Semantic Analyzer in this research adapts syntax-driven semantic analysis technique. This technique has already developed in Larasati's research using different components[4]. Syntax-driven semantic analysis is done by attaching semantic rule into its associated syntactic rule to produce syntactic representation [12]. The final result is then processed by λ-reduction [12].Table below shows some syntactic rules and their associated semantic rule.

Table 3. Syntactic Rule and Its Associated Semantic Rule [5]

Syntactic Rule (BNF)	Semantic Rule
<S> ::= <NP><VP>	<VP>.sem(X)(Y) ∧<NP>.sem(Y)
<S> ::= <NP><VP><PP>	<VP>.sem(X)(Y) ∧<NP>.sem(Y) ∧<PP>.sem(X)
<S> ::= <*PRP><VP>	<VP>.sem(X)(Y) ∧<*PRP>.sem(Y)
<NP> ::= <*NN>	<*NN>.sem
<NP> ::= <*NN><PP>	<NP>.sem(X) ∧<PP>.sem(X)
<NP> ::= <*NN><*JJ>	<*NN>.sem(X) ∧<*JJ>.sem(X)
<VP> ::= <*VBT><NP>	<*VBT>.sem(X)(Y)(Z) ∧<NP>.sem(Z)
<VP> ::= <*VBI>	<*VBI>.sem
<VP> ::= <VP><NP>	<VP>.sem(X)(Y) ∧<NP>.sem(Y)
<PP> ::= <*IN><NP>	<*IN>.sem(X)(Y) ∧<NP>.sem(Y)
<ADJP> ::= <*JJ>	<*JJ>.sem
<ADVP> ::= <*RB>	<*RB>.sem

D. Transforming First Order Logic into Mind Map Symbol

In order to have the final Mind Map result, the next component should transform the original of first order logic into the needed representation for the Mind Map visualization. The example of first order logic result and the needed representation for sentence "Kartini lahir di Jepara" are shown in Figure 12.

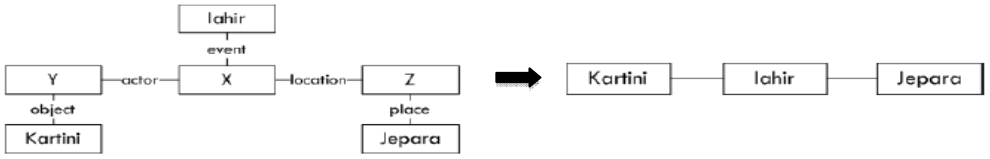

Figure 12. The First Order Logic Result and Its Needed Representation for Mind Map Visualization

To have the needed presentation which consists of object and relation, the variable of X, Y and Z are replaced by its related value of "lahir", "Kartini" and "Jepara" respectively. To make relation among sentences, the same objects are joined into one object.

E. Mind Map Visualization Editor

1). Mind Map Object Generator

Mind Map visualization was done by using several Mind Map rules such as drawing the main concept in the center of the figure with branches related with the center. We analyzed that this can be done by using the radial drawing method [19] where the root as the drawing center is the main concept while the branches are the entities related with the main concept. Figure 13 shows a non-tree graph example of the entities yielded from several Indonesian sentences.

Figure 13. The Object Graph for "Kartini lahir di Jepara. Kartini tinggal di Jepara."

The radial drawing method is a variation of layering drawing method [19]. The idea is to transform a tree representation into radial. The illustration is shown in Figure 14. Here, the red circle represents the main concept of the sentence which is placed at the center of the figure. The main concept is chosen from the concept with the most relations compared to other concept. The concepts related directly to the main concept are placed on the second layer, and so on.

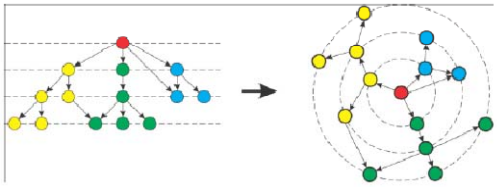

Figure 14. The Radial Drawing Method

2). Interaction in Mind Map Editor

The automatic generator usually doesn't yield accurate result, which means that to have the application can be used for real sentences, the generated Mind Map should be able to be modified easily. To cope with these needs, we also build the Mind Map editor. We defined that several problems that should be handled in the editor include: entity position, entity content, relation content, object property such as color and size, adding image, curvature of line, and the structure including the addition or deletion of entity or relation.

4. Experiments

Here, we will show the experiments for each component employed in the Indonesian Mind Map generator. Each experiment is described in following section.

A. Indonesian POS tagger using HMM

The complete experiments on POS Tagger are available on [6]. The training data is about 12000 words and the testing data is about 3000 words. The experimental result for 15% OOV words is available in Table below. There are three types of affix tree: 1) prefix tree; 2) suffix tree; 3) prefix and suffix tree. Even though the best result was achieved by using the prefix and suffix tree, but in several configurations, the prefix tree gave a slight higher accuracy than the prefix-suffix tree. We concluded that the prefix gives better prediction than the suffix on the POS tag for Indonesian language. Another conclusion is for the n-gram used in the configuration. Almost all configuration of bigram gave higher accuracy result than the trigram, which yielded a conclusion that to predict POS tag for Indonesian language, bigram is more suitable than trigram. Another conclusion is that using the succeeding POS tag doesn't give much improvement, furthermore it gives lower accuracy for some cases, therefore our conclusion is that the succeeding POS tag doesn't have important role in defining POS tag for Indonesian language.

Table 4. Experimental Result for Indonesian POS Tagger using HMM on 15% OOV Testing Data[6]

NO	*Configuration*	*Affix Tree Configuration*		
		PREFIX	***SUFFIX***	***PRE-SUFF***
1	Baseline	*90.65% (99.42%/42.34%)*		
2	Bigram	95.67% (99.43%/75.00%)	94.32% (99.39%/66.44%)	95.36% (99.43%/72.97%)
3	Trigram	95.29% (99.18%/73.87%)	94.29% (99.22%/67.12%)	95.01% (99.22%/71.85%)
4	Bigram+succeeding POS	95.57% (99.43%/74.32%)	94.56% (99.35%/68.24%)	95.36% (99.43%/72.97%)
5	Trigram+succeeding POS	94.94% (99.02%/72.52%)	94.04% (99.06%/66.44%)	94.70% (99.02%/70.95%)
6	Bigram+Lexicon	96.30% (99.43%/79.05%)	95.01% (99.43%/70.72%)	96.23% (99.43%/78.60%)
7	Trigram+Lexicon	95.98% (99.18%/78.38%)	94.94% (99.26%/71.17%)	95.95% (99.26%/77.70%)
8	Bigram+succ+Lexicon	96.36% (99.43%/79.50%)	95.36% (99.43%/72.97%)	96.50% (99.43%/80.41%)
9	Trigram+succ+Lexicon	95.78% (99.02%/77.93%)	95.08% (99.06%/73.20%)	95.91% (99.06%/78.60%)

B. Indonesian Syntactic Parser using CYK and Earley

The experimental data consists of 100 sentences for training data and 36 sentences for testing data. The training and testing data were labeled using modified English parser which results were then checked manually. The data consists of several patterns of single sentence and compound sentence. The sentences contain patterns of simple sentence (1 clause), compound sentence and complex sentence (Subject and Object subordinate). The comparisons between the CYK and Earley algorithm results are shown in Table below. Even though using the Earley algorithm gave better result than CYK in finding the correct parse tree among the correct candidate, but in selecting the one best parse tree, CYK gave better result than Earley algorithm. In general, the syntactic parser of Indonesia still couldn't give good accuracy since several reasons such as the corpus size and that there is no enhancement on the basic parsing algorithm employed here. Another reason is that the complexity of Indonesia grammar where there are cases where the phrase limit is not clear and there are cases where the sentence predicate doesn't exist, unlike English sentence.

Table 5. Experimental Result for Indonesian Syntactic Parser using CYK and Earley Algorithms on 100 Sentence of Testing Data

	CYK	Earley
Correct parse tree is found as one of the correct candidate	47.22%	53.85%
Correct parse tree is selected as the best parse tree found	47.22%	38.89%

C. Semantic Analyzer from CYK Syntactic Parser

The Semantic Analyzer component uses the result of Indonesian probabilistic parser with Cocke-Younger-Kasami (CYK) Algorithm. Table below shows the accuracy of each component in the Semantic Analyzer.

Table 6. Experimental Result for Indonesian Semantic Analyzer of CYK Syntactic Parser[5]

Component	Accuracy
Lexical semantic attachment	88.89%
Reference resolution	66.67%
Semantic Analyzer	62.50%

We found that in the experiments, there are several sentence structures that cannot be handled by system, such as below[5]:

1) Sentence with two nouns after verb or predicate, for example "Ibu/PN membelikan/VBT adik/NN baju/NN" (Mother bought a dress for sister), where "adik" (sister) and "baju" (a dress) should be treated as two different phrases but the parser falsely treated them as one noun phrase similar with "baju/NN adik/NN" (sister dress).
2) Sentence with non-noun subject, such as the usage for gerund.
3) Sentence with non-verb predicate, for example "Orang/NN itu/DET tinggi/ADJ" (That person is tall), where the predicate only consists of one adjective ("tinggi"), different with English sentence that always has a verb as the predicate such as "is" instead of "tall".
4) Sentence with two verbs in form of simple sentence or compound sentence, for example "Pemerintah/NN setuju/VBT menaikkan/VBT harga/NN listrik/NN" (The government is agree to raise the electricity price), where the predicate consists of two verb ("setuju"/agree and "menaikkan"/raise) and should be treated as different phrase, different with English sentence where the phrase limit is clear by using the word "to" between "agree" and "raise".

D. Mind Map Generator

1) Experiment to Evaluate the FOL-Semantic Network Transformation

The original text consists of 34 sentences, while the modified text consists of 59 sentences. Here, the complex sentences are modified into simple sentences. Even though the sentences are modified into simple sentences, still not all texts can be processed since the limited rules and training data available in the Indonesian POS Tagger, Syntactic Parser and Semantic Analyzer. For the original text, only 17 sentences that can be processed from 34 sentences, and only 2 sentences were processed correctly, which gives accuracy of around 6%. For the modified text, there are 47 sentences can be processed from 59 sentences, and 33 text were processed correctly, which gives accuracy of 56%. Mainly the error is caused by the Syntactic Parser, while the error caused by the transformation is only 1 sentence from both texts.

2) Experiment to Evaluate Mind Map Drawing Result

Here, we asked 5 respondents to evaluate the legibility of the resulted Mind Map visualization. There are two result types: (1) the original automatic one, resulted by the system and (2) the modified one.

As the result, there are 48% of respondents said that the original drawing is readable and easy to understand. As for the modified one, there are 96% respondents said the drawing is readable and easy to understand. We analyzed that there are some unimportant words from the original sentence that makes the result is difficult to be understood. Another drawbacks are the color and the main idea focus.

5. Conclusion

We have conducted research to investigate the transformation of Indonesian text sentence into Mind Map representation. The system includes several Indonesian natural language understanding tools such as Indonesian POS Tagger, Indonesian Syntactic Parser, Indonesian Semantic Analyzer and Mind Map generator. We transform the input text into its semantic representation in first order logic by using Indonesian natural language understanding tools. The techniques used in the natural language understanding tools for Indonesia are adapted with availability of Indonesian language resources. For Indonesian POS Tagger, a decision tree is used to handle to empty score of emission probability on the HMM method. This is due to the small size of the POS Tagged Indonesian corpus employed in this research. For the Indonesian Syntactic Parser, the idea is to use the Probabilistic CFG and provide the Indonesian syntactic tagged corpus for the training and testing data. In the Semantic Analyzer, since there is no available resource on the semantic information of each lexicon, then we define some rules to

attach the lexical semantic information to the sentence. The semantic representation is then transformed automatically into Mind Map object and visualization using several defined rules and radial drawing method. The experimental results showed that since there are weaknesses in the Indonesian natural language understanding tools, the best result of Mind Map generator can only be done for simple and efficient Indonesian sentence. In future works, we will work on the improvement of each Indonesian natural language understanding tool and additional process of Mind Map generator to filter the unnecessary words.

References

[1] T. Buzan,"Buku Pintar Mind Map", Jakarta, PT. Gramedia Pustaka Utama, 2005

[2] M. Abdeen, R. El-Sahan, A. Ismaeil, S. El-Harouny, M.Shalaby, M. C. E. Yagoub, "Direct Automatic Generation of Mind Maps from Text with M2Gen", *in Proceeding of IEEE Toronto International Conference Science and Technology for Humanity*, pp. 95-99, Toronto, Canada, 2009

[3] C. Brucks, C. Schommer, "Assembling Actor-based Mind-Maps from Text Streams", *Master Thesis, University of Luxembourg, Department of Computer Science and Communication*, 2008

[4] S.D. Larasati, R. Manurung, "Towards a Semantic Analysis of Bahasa Indonesia for Question Answering". *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics, Melbourne, Australia*, 2007

[5] F. Ferdian, A. Purwarianti, "Implementation of Semantic Analyzer in Indonesian Text-Understanding Evaluation System"., *in Proceedings of IEEE International Conference on Computational Intelligence and Cybernetics, Bali*, 2012

[6] A.F. Wicaksono, A. Purwarianti, "HMM based Part of Speech Tagger for Bahasa Indonesia", *Fourth International MALINDO Workshop, Jakarta*, 2010

[7] F. Pisceldo, R. Manurung, M.Adriani, "Probabilistic Part-of-Speech Tagging for bahasa Indonesia", *Third International MALINDO Workshop, colocated event ACL-IJCNLP* 2009, Singapore, August 1, 2009

[8] M. Adriani, H. Riza, "Research Report on Local Language Computing: Development of Indonesian Language Resources and Translation System", Ref. No: PANL10n/Admn/RR/001, PAN Localization Project, 2008

[9] N. Nguyen, Y. Guo, "Comparisons of Sequence Labeling Algorithms and Extensions". *International Conference on Machine Learning, Corvallis, USA*, 2007

[10] H. Schmid, "Probabilistic Part-of-Speech Tagging using Decision Tree", *Proceedings of International Conference on New Methods in Language Processing, Manchester*, September 1994

[11] H. Schmid, "Improvements in Part-of-Speech Tagging with an Application to German". *Proceedings of the ACL SIGDAT-Workshop, Dublin, Ireland*, March 1995

[12] D. Jurafsky, J. H. Martin, "Speech and Language Processing", 2nd Edition, Prentice Hall, 2009

[13] J. Earley, "An Efficient Context-Free Parsing Algorithm", Communications of the ACM, Volume 13, Number 2, 1970

[14] A. Stolcke, "An Efficient Probabilistic Context-Free Parsing Algorithm that Computes Prefix Probabilities", *Journal of Computational Linguistics*, Volume 21, 1995

[15] R.J. Brachman, H.J. Levesque, "Knowledge Representation and Reasoning", Elsevier, 2004

[16] M. Bates, "Models of Natural Language Understanding", BBN Systems and Technologies, 1993

[17] J. Hobbs, "Ontological Promiscuity", Associational for Computational Linguistics, Chicago, USA, 1985

[18] B.V. Durme, et.al, "Towards light semantic processing for question answering", *Associational for Computational Linguistics, Edmonton*, Canada, 2003

[19] G. Batista, P. Eader, R. Tamassia, I. Tollis, "Graph Drawing: Algorithm for the Visualization of Graphs", Prentice Hall, 1999

Ayu Purwarianti. She was graduated from her bachelor and master degree at Informatics Program, Bandung Institute of Technology. She got her doctoral degree from Toyohashi University of Technology, Japan. Since 2008, she has become a lecturer at School of Electrical Engineering and Informatics, Bandung Institute of Technology, Indonesia. Her research interest is on computational linguistics, mainly on Indonesian natural language processing and Indonesian text mining. She is now active as the education officer at IEEE Indonesia.

Athia Saelan. She was graduated from her bachelor degree at Informatics Program, Bandung Institute of Technology on 2012. She is now working as software developer at PT. Akhdani Reka Solusi, Indonesia. She has research interest in artificial intelligence.

Irfan Afif. He was graduated from his bachelor degree at Informatics Program, Bandung Institute of Technology (ITB) on 2011. When he was graduated from ITB, he founded a game developer, namely Whappa Games, with his friends. Currently, he is working as senior software developer for android at Radya Labs Technology. His research interest is on android programming and algorithm.

Filman Ferdian. He was graduated from his bachelor degree at Informatics Program, Bandung Institute of Technology on 2012. He is now working as system analystat McKinsey & Company, Indonesia. His research interest is on artificial intelligent.

Alfan Farizki Wicaksono. He was graduated from his bachelor degree at Informatics Program, Bandung Institute of Technology on 2010. He got his master degree at KAIST, Korea. He is now working as a lecturer at Faculty of Computer Science, University of Indonesia. His research interest mainly focuses on text mining, information retrieval, and topic modeling.