Journal of Theoretical and Applied Information Technology
29th February 2016. Vol.84. No.3
© 2005 - 2016 JATIT & LLS. All rights reserved·

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DESIGNING AND IMPLEMENTING PARSING FOR AMBIGUOUS SENTENCES IN INDONESIAN LANGUAGE

[1]DEWI SOYUSIAWATY, [2]EKO ARIBOWO
[1]Informatics Department, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
[21]Informatics Department, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
E-mail: [1]dewi.soyusiawaty@tif.uad.ac.id, [2]ekoab@uad.ac.id

ABSTRACT

In the daily life there are a lot of ambiguous words or sentences. For instance, in Indonesian language the word ‘apel’ has two different meanings if it is placed in the sentence. It may mean 1) to hold a ceremony or 2) a name of a fruit. Another example is the word ‘tahu’ that may mean 1) understand or know or 2) a name of a food from soybean. The ambiguous sentences become a matter in the translation application or dictionary if the application has no facility to detect the sentence with some different meanings. As a result, it will influence the translation result. The word ‘tahu’ in the sentence ‘saya ingin tahu’, if it is translated into one of regional language such as Javanese language, it may result ‘kula kepengen tahu’ (I want a food from soybean) or ‘kula kepengen weruh’ (I want to know). It depends on what ‘tahu’ means. This research discusses the role of parsing as the sentence breaker in identifying the ambiguous sentences, so the real meaning of the sentence can be obtained and the congruence between structure pattern of a sentence that is inputted and grammar rules saved can be acknowledged.
The research started by collecting Indonesian language grammar rule data consisting of clause structure forming a sentence, phrase that is two or more words structure, and detail clause pattern structure including ambiguous phrases and vocabularies list. Then, flowchart for parsing is designed in some stages those are obtaining pattern and checking the phrase, implementing and evaluating the application.
The research resulted the sentence parsing flowchart, application for ambiguous sentences checking and to know the structure congruence between the input sentence and the rules.

Keywords: *Parsing, Sentence, Ambiguous, Clause, Phrase*

1. INTRODUCTION

A. The Ambiguity in the Natural Language

Many translation applications have not been able to detect the ambiguous sentences. For instance, the word ‘kali’ has some meanings, (1) river, (2) frequency, (3) mathematic operation. The word ‘tahu’ may mean (1) to know, understand, (2) a name of a food from soybean. The word ‘apel’ may mean (1) a name of a fruit, (2) to hold a ceremony. Most of the systems only translate based on the sentence inputted. Whereas, before translating into the target language, some information needs to be obtained to produce the correct translation. [1]

B. Parsing

Process analysis stage syntactic parsing is useful for checking the order of appearance of the token or the appropriate and whether or not a sentence with grammatical syntax (grammar). Parsing method is divided into two, namely top down parsing and bottom up parsing. Phase sequence in the compiler is Lexical analysis or parsing, Semantic analysis and code generation. A unit in a language (a "word") is called "token". A token is usually a word or symbol. Literal something that cannot be broken again called terminal. Parsing (syntactic analysis) is a process for analyzing the token to determine the structure of the grammar. Parsing process usually consists of two parts, the first being that combines character by character to form a token (usually performed by a part called a scanner or Lexar), and the second part is that determines whether the tokens that meet the grammar (parser). Table 1introduces the $L1$ grammar, which consists of the $L0$ grammar with a few additional rules. The goal of a parsing search is to find all the trees whose root is the start symbol S and which cover exactly the words in the input. [4]

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1 : L1 Grammar

S → NP VP	Det → that |this|a
S → Aux NP VP	Noun → book|flight
S → VP	Verb → book|include
NP → Pronoun	Pronoun → I|she|me
NP → Proper - Noun	Proper-Noun → Houston
NP → Det Nominal	Aux → does
Nominal → Noun	Prep → from|to|on
Nominal → Nominal Noun	
Nominal → Nominal PP	
VP → Verb	
VP → Verb NP	
VP → Verb NP PP	
VP → Verb PP	
VP → VP PP	
PP → Preposition NP	

B.1 Top Down Parsing

A top-down parser searches for a parse tree by trying to build from the root node *S* down to the leaves. It builds all possible trees in parallel. The algorithm starts by assuming the input can be derived by the designated start symbol *S*. The next step is to find the tops of all trees which can start with *S*, by looking for all the grammar rules with *S* on the left-hand side. In the grammar in Table 1, there are three rules that expand *S*, so the second ply, or level, of the search space in Figure 1 has three partial trees.[3]

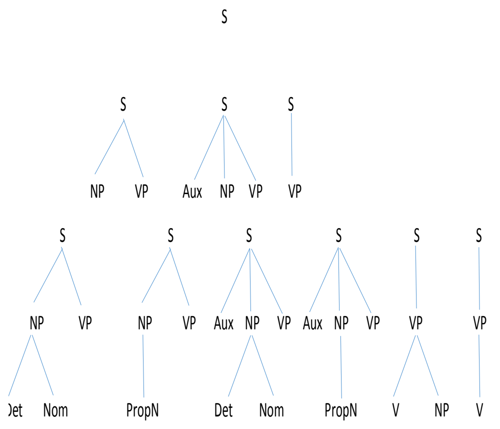


Figure 1: An Expanding Top Down Search Space

B.2 Bottom Up Parsing

The parser starts with the words of the input, and tries to build trees from the words up, again by applying rules from the grammar one at a time. The parse is successful if the parser succeeds in building a tree rooted in the start symbol *S* that covers all of the input. Figure 2 shows the bottom-up search space, beginning with the sentence *Book that flight*. The parser begins by looking up each input word in the lexicon and building three partial trees with the part-of-speech for each word. But the word *book* is ambiguous; it can be a noun or a verb. Thus the parser must consider two possible sets of trees. The first two plies in Figure 2 show this initial bifurcation of the search space.

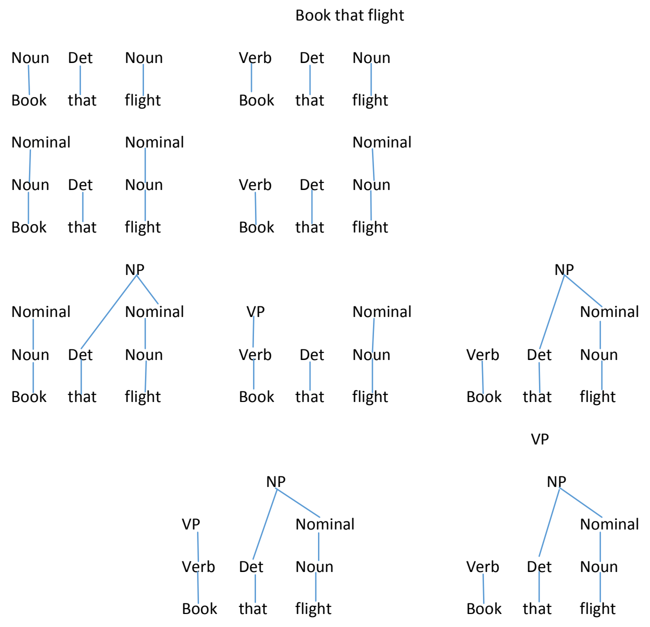


Figure 2: An Expanding Bottom Up Search Space

C. CFG

Context Free Grammar is used to explain the syntax of a language. Grammar is a form of human’s natural language used in the daily communication. Every grammar rule has syntactic structure that can be described by a grammar.

The components of CFG:

1. A group of token namely terminal
2. A group of non-terminal
3. A group of production rules
4. First symbol

In the CFG it needs to understand the following terms related to grammar in order to differentiate one symbol to another.

a. The following symbols are terminal
   1) Lowercase letter, in the first alphabet letters such as a, b, or c
   2) Operator symbols such as @, #, -, etc
   3) Number 0, 1, 2, etc
b. The following symbols are non-terminal
   Capital letter, from alphabet, A, B and C
   The first symbol

c. A lowercase letter of the end of alphabet such as u, v,...z represents a group of terminal
d. A lowercase Greek letter such as α, β, μ as an example represents a group of grammar symbol
e. If no another statement, the left part of production is the first symbol. [5]

D. Indonesian Language Grammar

The sentence constructor structure physically is the clause. The clause is a lingual unit consisting of minimal subject (S) and predicate (P). The other clause constructor elements are object (O), complement/pelengkap (Comp/Pel) and adverb/ keterangan (Adv/K). The following Table 2 is the clause structure pattern in Indonesian language that is commonly used [7]

Table 2: Clause Structure

No	Pattern Clause	Example of sentences
1	S-P	Dia belajar
2	S-P-O	Adik makan roti
3	S-P-Pel	Aku belajar menari
4	S-P-O1-O2	Kakek membelikan adik sepeda baru
5	S-P-O-K	Ia menendang bola ke atas atap rumah
6	S-P-Pel-K	Aku berenang gaya katak di Umbul Tirto kemarin
7	S-P-O1-O2-K	Kakek membelikan adik sepeda baru kemarin

A phrase is a grammatical unit consisting of two or more words which do not cross the clause function. The following Table 3 shows the type of phrase, pattern and the example.

Table 3: Types of Phrase

No	Type	Pattern	Example
1	Frase Nomina/FN/ Noun Phrase	N – N	Orang itu
		N – V	Gadis cantik
		N – Numbers	Kucing dua
		N + Desc	Koran kemarin pagi
		N + FD	Beras dari Delanggu
		Numbers + N	Enam penjahat
		Si + N	Si Ahmad
		Yang + N	Yang ini
		Yang + V	Yang bertopi
		Yang + Desc	Yang sekarang
		Yang + Numbers	Yang tiga buah
		Yang + FD	Yang ke Surabaya
2	Frase Verba/FVVerb Phrase	Verba	Sedang makan
			Cantik sekali
			Makan dan minum
3	Frase Keterangan/FV	Desc + N	Besok pagi
			Kemarin sore
			Tadi siang
4	Frase Bilangan/Fbil /Numbers Phrase	Numbers + unit	Dua ekor
			Lima buah
			Seratus orang

From the clause pattern and the phrase type, the basic sentence pattern can be combined. It consists of the functions which are filled by the phrase that consists of the word categories. The sentence structure can be shown on the Table 4 with the following compositions:

Table 4: Clause Pattern

NO	Pattern				
1	S	P			
	N/FN	N/FN			
		V/FV			
		Bil/FbIL			
		FD			
2	S	P	O		
	N/FN	V/FV	N/FN		
3	S	P	P (Pelengkap)		
	N/FN	V/FV	N/FN		
			BIL/FBIL		
			V/FV		
4	S	P	K		
	N/FN	N/FN	K/FKet		
		V/FV	FD		
		BIL/FBIL			
5	S	P	O	K	
	N/FN	V/FV	N/FN	K/FKet	
				FD	
6	S	P	O1	02	
	N/FN	V/FV	N/FN	N/FN	
7	S	P	O1	O2	K
	N/FN	V/FV	N/FN	N/FN	KET/FKet
					FD

2. DISCUSSION

From the structure data of the clause, phrase and clause patterns, then the hierarchy of grammar can be arranged as follows :

A. CFG

Formally, a grammar consists of four elements:

1. T terminal finite set consisting of all symbols used to state the sentences in Indonesian language
   !"#$ %&'()*+,-./ 0123456789 : ;<=> ?@ ABCDEFGHIJKLMNOPQRSTUVW XYZ[\]^_`abcdefghijklmnopqrstuvwxyz{¦}~¢£¤ ¥¦§¨©ª«¬®°±²³´µ•
2. N Non-terminal finite set is a symbol to symbolize a sentence, phrase and word group. Table 5 consists of the list of the sentence symbol from the above clause structure, phrase and word group.

Table 5: Non Terminal (N)

No	Non Terminal	Symbol
1	Sentence/Kalimat	Kal
2	Subjek	S
3	Predicate	P
4	Object	O
5	Complement/Pelengkap	PEL
6	Noun Phrase/Frase Nomina	FN
7	Verb Phrase/Frase Verba	FV
8	Numbers Phrase/Frase Bilangan	FBIL
9	Front Phrase/Frase Depan	FD
10	Adverb Phrase/Frase Keterangan	FKET
11	Nomina	N
12	Verb	V
13	Adjective	Adj
14	Adverb	Ket
15	Number/Bilangan	BIL
16	Question/Tanya	Tanya
17	Conjunction	Hub
18	Preposition/Kata Depan	Depan
19	etc	

3. P Production Rule Finite Set. The production rule in this research is a combination of clause pattern and the phrase. The following Table 6 shows the rule combination to produce the correct sentence from Table 3, Table 4 and Table 5.

Table 6: Production Rule

No	Production Rule
1	<Kal> = <S><P>
2	<Kal> = <S><P><O>
3	<Kal>= <S><P><PEL>
4	<Kal>=<S><P><O1><O2>
5	<Kal>= <S><P><O><K>
6	<Kal>=<S><P><PEL><K>
7	<Kal>=<S><P><O1><O2><K>
8	<S>= <N> | <FN>
9	<P>=<N> | <FN>
10	<P>=<V>|<FV>
11	<P>=<Bil> | <Fbil>
12	<P>=<FD>
13	<O>=<N>|<FN>
14	<K>=<KET>|<FKET>
15	<K>=<FD>
16	<PEL>=<N>|<FN>
17	<PEL>=<V>|<FV>
18	<PEL>=<BIL>|<FBIL>
20	<FN>=<N><N>
21	<FN>=<N><V>
22	<FN>=<N><BIL>
23	<FN>=<N><KET>
24	<FN>=<N><FD>
25	<FV>=<V><TAMBAH>
26	etc

4. The S ε N first symbol
<Kal>

The example of bottom up parsing process of the sentence ‘Saya makan apel’ as follows:
In the parsing process number 1 and 2, ‘apel’ functions as ‘V’ that is to hold a ceremony and the suitable pattern structures are S – P and S – P – Pel. The parsing process number 3, 4, and 5, ‘apel’ functions as ‘N’ that is a fruit and the suitable pattern structures are S – P, S – P – O and S – P – Pel.

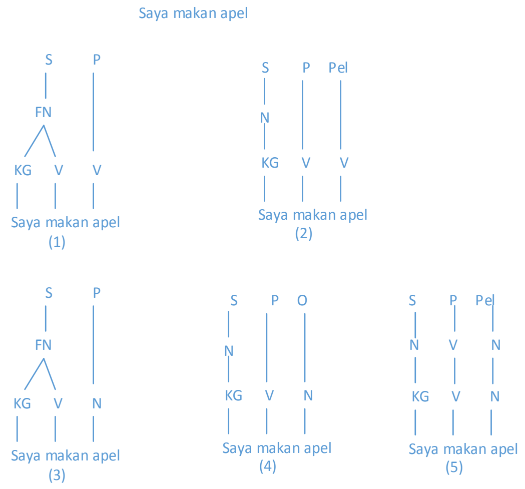


Figure 3: Bottom Up Parsing Process Of The Sentence ‘Saya Makan Apel’

B. The Ambiguous Sentence Parsing

The stages of breaking the sentences in general to detect the ambiguous sentence as follows:

1) The process of breaking an input text into sentences and words. This process is fully managed by the common string processing function in every programming language. The sentence identification process can be acknowledged by the existence of full stop character (“.”), question mark (“?”), exclamation mark (“!”), and different line position. The word identification process can be acknowledged by the space character.
2) Every sentence that is obtained will be grouped into variables which have words as the constructor.
3) The words obtained will be checked with the database to get the detail information such as word type, translation, and other available information. One word may have more than one word types.
4) Every type of a word in a sentence will be rearranged. Since one word may have more than one word type, the structure of word type in a sentence can also give more than one combination of word type structure.
5) The next process is the parsing process of the word type structure towards the clause pattern structure. The method used in this process is top down parsing. All clause patterns registered will be checked with the word type structure

obtained. This process will also consider the possibility of the clause pattern that is structured from the phrase by breaking the phrase into the smallest component which is the word type.

6) The result of the clause pattern parsing process will then be put in a variable by maintaining the word type information suitable to the clause pattern. The system will choose the word structure based on the word type that meets the clause pattern. If there are more than one clause pattern, the system will choose the first suitable one.

C. Parsing as the Stage for Checking the Syntax Or the Sentence Structure

The sentence structure data is shown in an array as follows:

```
Array [
0 => [
Indonesian_word => ‘word_1’,
transl_word => [
0 => transl_word _1 // transl_1
n => transl_word _n // alternate translate to n
                    ],
Word_group => [
0 => Word_group _1 // word group of transl_1
n => Word_group _n // word group of translation to
n
                    ]
          ]
n => [
Indonesian_word => ‘word_n’,
Transl_word => [
0 => Transl_word _1
n => Transl_word _n
                    ],
Word_group => [
0 => word_group _1
 n => word_group _n
                    ]
          ]

]
```

The example of the sentence ‘saya makan apel’. The word ‘apel’ has two translations those are a fruit (N) and to hold a ceremony (V), so the translation data obtained is:

```
[
0 => [
Indonesian_word => saya
transl_word      => [0 => kula]
word_group       => [0 => K_GANTI]
          ],
1 => [
Indonesian_word => makan
transl_word      => [0 => mangan]
word_group       => [0 => V]
          ],
2 => [
Indonesian_word => apel
transl_word      => [0 => apel,1 => apel]
word_group       => [0 => V,1 => N]
          ],
]
```

The process of checking syntax has some stages as follows:

1) The process to get the pattern

The process is started by reading the sentence data consisting of Indonesian word data, translation and word group. Every word group in a sentence is taken and put in an array.

The result of this step is:

word_group =[[0 => K_GANTI],[0 => V],[0 => V,1 => N]]

After the process of taking word group, pattern formation is done by combining all data group data as follows:

1. Pattern[0] = K_Ganti
2. Pattern[0] = K_Ganti-V
3. Pattern[0] = K_Ganti-V-V
   Pattern[1] = K_Ganti-V-N

In the end of the process, two pattern combinations will be obtained, K_Ganti-V-V and K_Ganti-V-N because there is one word having two word groups. Figure 4 shows the flowchart to get the pattern.

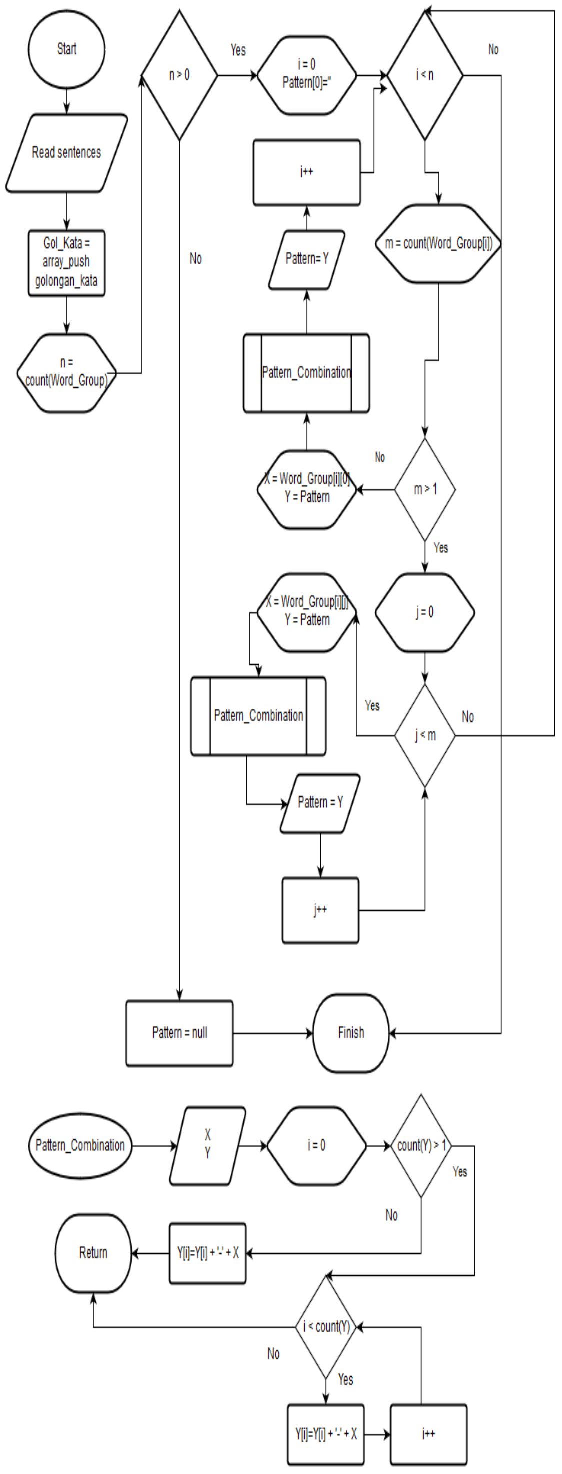


Figure 4 : Flowchart To Get The Pattern

2) The checking of maximum number of phrase constructors

The phrase can be formed by several word groups. The phrase data is obtained from the database. If a sentence only consists of two words, the phrase checking process may only be done till two constructor words of the phrase. Therefore, the possible result is that in the sentence there are two phrases with one word group constructor and one phrase with two word group constructors. The process is started by reading the phrase data from the database and the number of the words in the sentence.

If in the database, the phrase data can be found, the maximal phrase constructor is one. If it is found, the maximal phrase constructor is searched. The last step is comparing the maximal phrase constructor and the number of words in a sentence and the least is taken.

The checking of this maximal phrase constructor is necessary to limit the process of composing the phrase from the word group and as the control, so the system doesn't mistake because the data received is smaller than the number of the phrase composer data (out of index). Figure 5 shows the flowchart of the checking of maximal number of phrase constructors.

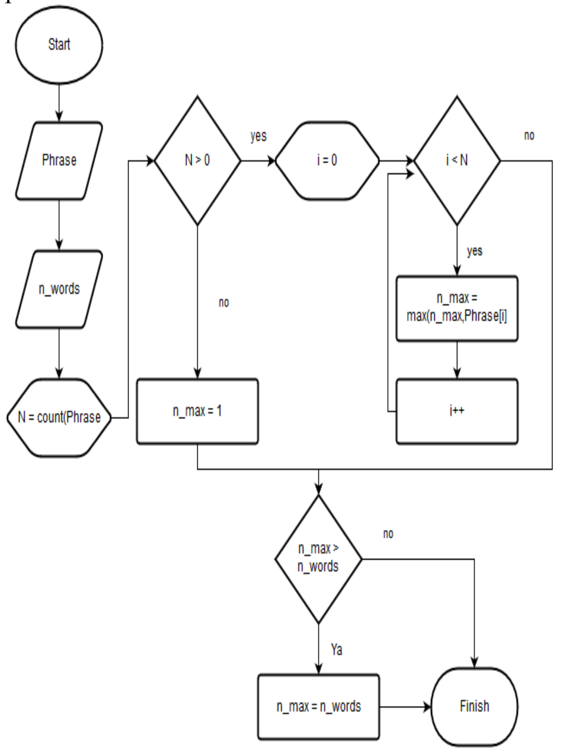


Figure 5: Flowchart Of The Checking Of The Maximal Number Of Phrase Constructor

The whole syntax checking stages are shown on the figure 6 as follows.

After the number of maximal phrase constructors and sentence pattern data are obtained, the system will replace the sequence of the available sentence pattern into phrase. The replacement process starts with the smallest number of phrase constructor which is one until the maximal one.

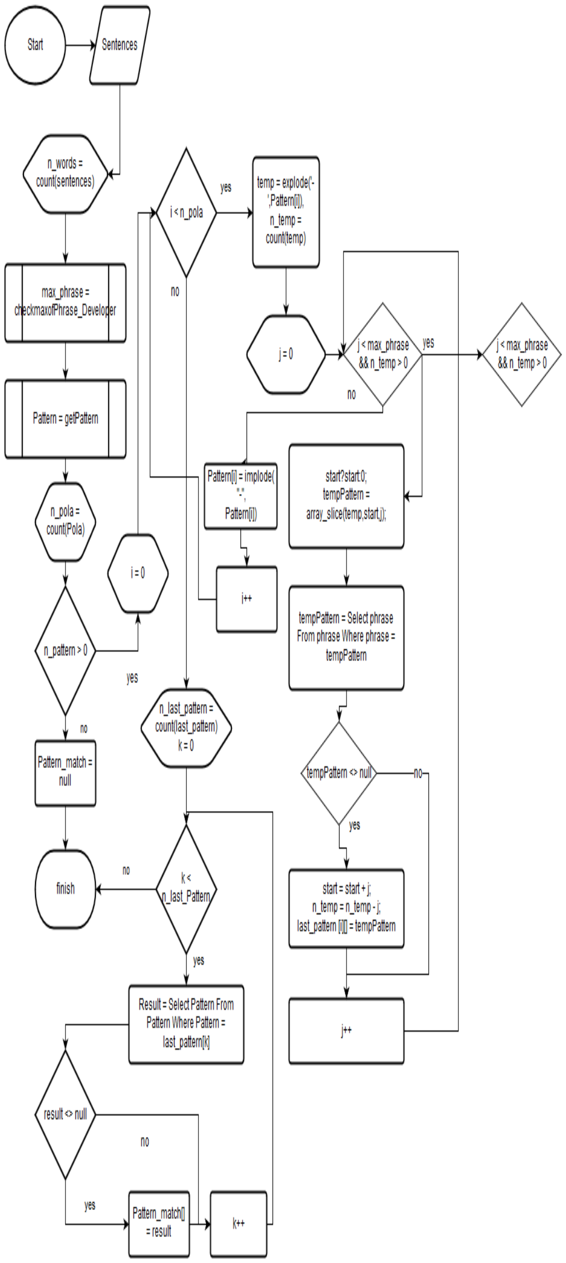
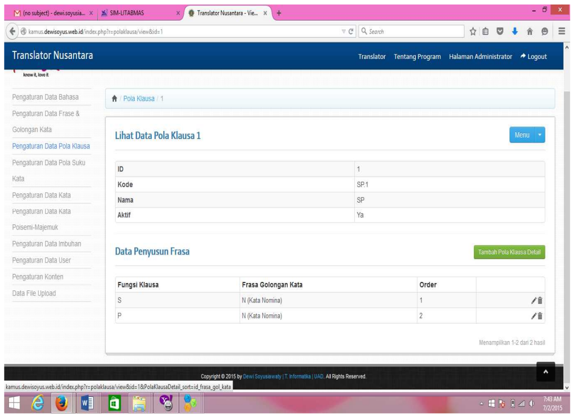

Start
Sentences
n_words = count(sentences)
max_phrase = checkmaxofPhrase_Developer
Pattern = getPattern
n_pola = count(Pola)
n_pattern > 0
yes
no
Pattern_match = null
finish
i = 0
i < n_pola
yes
no
temp = explode(' ',Pattern[i]), n_temp = count(temp)
j = 0
j < max_phrase && n_temp > 0
yes
no
j < max_phrase && n_temp > 0
Pattern[i] = implode(' ', Pattern[i])
i++
start?start:0; tempPattern = array_slice(temp,start,j);
tempPattern = Select phrase From phrase Where phrase = tempPattern
tempPattern <> null
yes
no
start = start + j; n_temp = n_temp - j; last_pattern [i][] = tempPattern
j++
n_last_pattern = count(last_pattern) k = 0
k < n_last_Pattern
yes
no
Result = Select Pattern From Pattern Where Pattern = last_pattern[k]
result <> null
yes
no
Pattern_match[] = result
k++

Figure 6: Flowchart of Syntax Checking

C. Application

The application on the application involves the word data storage including the word type, phrase, and clause pattern data. The phrase is used to store the information of all word type combination structures those can form the phrase. The clause pattern that is applied in this application involves all compositions on the table 4. Every clause pattern is explained according to the composition of the structure both a single word and a phrase. The example of the clause pattern storage can be seen on the figure 7 and figure 8.

Figure 7 shows the clause pattern application of Table 6 with the production rules number 1, 8 and 9, those are by setting S placed by N and P placed by N.


(no subject) - dewi.soyusia...
SIM-LITABMAS
Translator Nusantara - Vie...
kamus.dewisoyus.web.id/index.php?r=polaklausa/view&id=1
Search
Translator Nusantara
Translator
Tentang Program
Halaman Administrator
Logout
know it, love it
Pengaturan Data Bahasa
Pengaturan Data Frase & Golongan Kata
Pengaturan Data Pola Klausa
Pengaturan Data Pola Suku Kata
Pengaturan Data Kata
Pengaturan Data Kata Poisemi-Majemuk
Pengaturan Data Imbuhan
Pengaturan Data User
Pengaturan Konten
Data File Upload
Pola Klausa / 1
Menu
Tambah Pola Klausa Detail
7:43 AM
7/2/2015

Lihat Data Pola Klausa 1

ID	1
Kode	SP.1
Nama	SP
Aktif	Ya

Data Penyusun Frasa

Fungsi Klausa	Frasa Golongan Kata	Order	
S	N (Kata Nomina)	1	
P	N (Kata Nomina)	2	

Menampilkan 1-2 dari 2 hasil

Copyright © 2015 by Dewi Soyusiawaty | T. Informatika | UAD. All Rights Reserved.

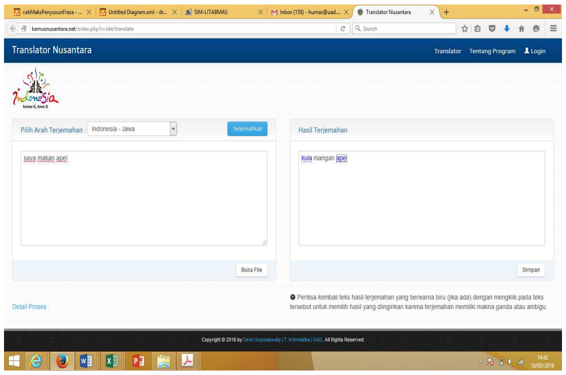
kamus.dewisoyus.web.id/index.php?r=polaklausa/view&id=1&PolaKlausaDetail_sort=id_frasa_gol_kata

Figure 7: Setting of S – P Clause, Consist of N – N

Figure 8 is the example of translation process by the system, where the word 'apel' has ambiguous meanings that may means a fruit and to hold a ceremony.

The system will choose the translation result of the word 'apel' based on the word type which is then checked with clause pattern combination received. If there are more than one suitable clause patterns, the system has to give options to edit the translation result produced as shown on the figure 9.

The new translation result produced will follow the user option.



Pilih Arah Terjemahan : Indonesia - Jawa

saya makan apel

Hasil Terjemahan

kula mangan apel

Detail Proses :

Periksa kembali teks hasil terjemahan yang berwarna biru (jika ada) dengan mengklik pada teks tersebut untuk memilih hasil yang diinginkan karena terjemahan memiliki makna ganda atau ambigu.

Copyright © 2016 by Dewi Soyusiawaty | T. Informatika | UAD. All Rights Reserved.

Figure 8: Translation of Ambiguous Sentence

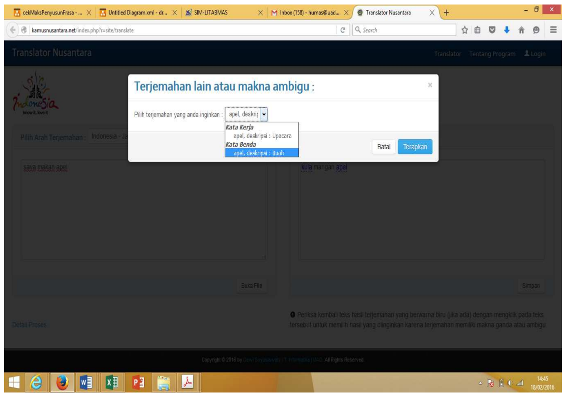
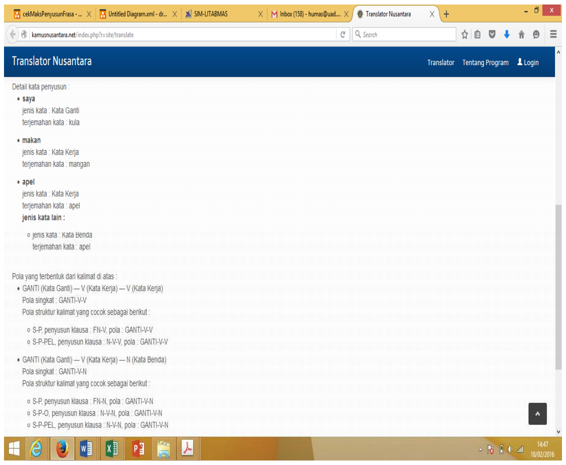

cekMaksPenyusunFrasa - ...
Untitled Diagram.xml - dr...
SIM-LITABMAS
Inbox (158) - humas@uad....
Translator Nusantara
kamusnusantara.net/index.php?r=site/translate
Search
Translator Nusantara
Translator
Tentang Program
Login
Terjemahan lain atau makna ambigu :
Pilih terjemahan yang anda inginkan :
apel, deskrip
Kata Kerja
apel, deskripsi : Upacara
Kata Benda
apel, deskripsi : Buah
Batal
Terapkan
Pilih Arah Terjemahan :
Indonesia - Ja
saya makan apel
kula mangan apel
Buka File
Simpan
Detail Proses
Periksa kembali teks hasil terjemahan yang berwarna biru (jika ada) dengan mengklik pada teks tersebut untuk memilih hasil yang diinginkan karena terjemahan memiliki makna ganda atau ambigu
14:45
18/02/2016

Figure 9: Post Editing in Ambiguous Sentences

The sentence parsing process is shown on the detail part of the process which is the stage of word tracing in the sentence and the process to find the sentence pattern in the database. Figure 10 shows some possibilities of suitable sentence patterns.


cekMaksPenyusunFrasa - ...
Untitled Diagram.xml - dr...
SIM-LITABMAS
Inbox (158) - humas@uad....
Translator Nusantara
kamusnusantara.net/index.php?r=site/translate
Search
Translator Nusantara
Translator
Tentang Program
Login
14:47
18/02/2016

Detail kata penyusun :

- **saya**
  jenis kata : Kata Ganti
  terjemahan kata : kula
- **makan**
  jenis kata : Kata Kerja
  terjemahan kata : mangan
- **apel**
  jenis kata : Kata Kerja
  terjemahan kata : apel
  jenis kata lain :
  - jenis kata : Kata Benda
    terjemahan kata : apel

Pola yang terbentuk dari kalimat di atas :

- GANTI (Kata Ganti) — V (Kata Kerja) — V (Kata Kerja)
  Pola singkat : GANTI-V-V
  Pola struktur kalimat yang cocok sebagai berikut :
  - S-P, penyusun klausa : FN-V, pola : GANTI-V-V
  - S-P-PEL, penyusun klausa : N-V-V, pola : GANTI-V-V
- GANTI (Kata Ganti) — V (Kata Kerja) — N (Kata Benda)
  Pola singkat : GANTI-V-N
  Pola struktur kalimat yang cocok sebagai berikut :
  - S-P, penyusun klausa : FN-N, pola : GANTI-V-N
  - S-P-O, penyusun klausa : N-V-N, pola : GANTI-V-N
  - S-P-PEL, penyusun klausa : N-V-N, pola : GANTI-V-N

Figure 10: Detail Process of Parsing

1) Pattern[0] = K_Ganti-V-V, is exploded with delimiter "-" so the data obtained is [K_Ganti, V, V], then the data is broken and arranged into a phrase:

   Phrase constructor 1 so the pattern obtained is pattern N [N,V,V]

   K_Ganti → N
   V → V
   V → V

   Phrase constructor 2 so the pattern obtained is pattern [FN,V]

   K_Ganti dan V → FN
   V → V

2) Pattern[1] = K_Ganti-V-N, is exploded with delimiter "-" so the data obtained is [K_Ganti, V, N], the data is broken and arranged into a phrase :

The phrase constructor 1 so the pattern obtained is patter N [N,V,N]

K_Ganti → N
V → V
N → N

The phrase constructor 2 so the pattern obtained is pattern [FN,N]

K_Ganti dan V → FN
N → N

After the process of phrase arrangement has finished, the data is then reunified with the function of implode delimiter "-" so there are 4 patterns obtained as follows.

```
Pola_Akhir = [
0 => [
        0 => N-V-V
        1 => FN-V
 ],
1 => [
        0 => N-V-N
        1 => FN-N
        ]
  ]
```

The 4 final patterns are then checked with the clause pattern data in the database and all suitable results are stored as the reference for first translation by the system. If the suitable patterns are more than one, the last suitable pattern is used to get the translation result.

Last_pattern [0] will suit the pattern S-P (FN-V) and S-P-Pel (N-V-V)

Last_pattern [1] will suit the pattern S-P (FN-N), S-P-O (N-V-N), and S-P-Pel (N-V-N)

If the translation taken is N-V-N, the sentence 'saya makan apel' will result 'kulo mangan apel' (N: a food)

The final translation result may be incorrect because of the last suitable pattern used. To solve the problem, all ambiguous meanings are displayed and the user can choose the translation they want. To solve the problem, it needs to add:

1) Calculation (the percentage of the phrase constructor decision probability). For example there are N and V, so the percentage calculation to choose between N and V or to be FN is given
2) The calculation on the clause pattern, for example there are two suitable clauses S-P-O and S-P-Pel. The system choose S-P-O because of the bigger probability calculation
3) The statistic of the word use through user interaction. For example the word 'apel' preceded by the word 'makan' has the bigger probability to choose because the statistic of user interaction to apel (food) compared to apel

(to hold a ceremony). The word ‘apel’ (to hold a ceremony) has the bigger probability if it is followed by adverb of time such as ‘apel pagi’.

3. CONCLUSION

a. The ambiguous sentence will produce parsing process with more than one clause pattern.
b. The detail clause pattern is needed to differentiate the ambiguous meaning of a sentence
c. In designing the application, the statistic storage of the word use and the translation have not been applied yet. The statistic of the use will help the translator system in giving the correct result particularly for the ambiguous sentences. The statistic of the use can give higher percentage to choose the correct clause pattern if the clause patterns is more than one. The more specific application in the clause pattern percentage is on the phrase because the probability in forming the phrase where the ambiguous meaning of the phrase could not easily be acknowledged compared to a single word.
d. The giving of correction suggestion towards the input text will produce more correct translation of a sentence, such as suggestion to add a comma or a conjunction to show that the pattern in the sentence is not equal.

4. ACKNOWLEDGMENT

The author would like to thank the Kemenristekdikti Indonesia which has provided research funding through research grants Penelitian Fundamental 2015 through a research agreement letter No. DIPA-023.04.1.673453/2015.

REFERENCES:

[1] Abdul Chaer, *Pengantar Semantik Bahasa Indonesia*, Jakarta: Rineka Cipta, 2009.

[2] Bogdan Patrut, Syntactic Analysis Based on Morphological Characteristic Features of The Romanian Language, International Journal on Natural Language Computing (IJNLC) Vol 1 No 4 December 2012.

[3] Daniel Jurafsky and James H Martin, “Speech and Language Processing, An Introduction to Natural Language Processing, Computational Lingustic and Speech Recognition”, Prentice Hall Series In Artificial Intelligence,1999.

[4]James Suciadi, Studi Analisis Metode Parsing dan Interpretasi Semantik pada Natural Language Processing, http://puslit.petra.ac.id/journals/informatics.

[5] Maman Abdurrohman, Syamsul Hadi and Dede Rohidin, “Pemeriksaan Tata Bahasa dalam Kalimat Bahasa Inggris Menggunakan Algoritma Left Corner Parsing”, Proceeding Seminar Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT2006), ISSN : 1411-6286, 23-24 Agustus 2006.

[6] Patterson, Don W.,1990, *Introduction to Artificial Intelligence and Expert Sistem*, Prentice Hall Internasional, Inc.

[7] Ramlan, ”Penggolongan Kata”, Yogyakarta : Andi Offset, 1985.

[8] Udaya Raj Dhungana, Subarya Shakna, Kabita Baral, Bharat Sharma, “Word Sense Disambiguation Using WSD Specific Wordnet of Polysemy Words”, International Journal on Natural Language Computing (IJNLC) Vol. 3 No. 4 August 2014.