A computational linguistic approach to natural language processing with applications to garden path sentences analysis

DU Jia-li
School of Foreign Languages/School of Literature
Ludong University/Communication University of China
Yantai/ Beijing, China

YU Ping-fang
School of Liberal Arts/Institute of Linguistics
Ludong University/Chinese Academy of Social Sciences
Yantai/ Beijing, China

***Abstract*— This paper discusses the computational parsing of GP sentences. By an approach of combining computational linguistic methods, e.g. CFG, ATN and BNF, we analyze the various syntactic structures of pre-grammatical, common, ambiguous and GP sentences. The evidence shows both ambiguous and GP sentences have lexical or syntactic crossings. Any choice of the crossing in ambiguous sentences can bring a full-parsed structure. In GP sentences, the probability-based choice is the cognitive prototype of parsing. Once the part-parsed priority structure is replaced by the full-parsed structure of low probability, the distinctive feature of backtracking appears. The computational analysis supports Pritchett's idea on processing breakdown of GP sentences.**

***Keywords*- Natural language processing; computational linguistics; context free grammar; Backus–Naur Form; garden path sentences.**

I. INTRODUCTION

The advent of the World Wide Web has greatly increased demand for natural language processing (NLP). NLP relates to human-computer interaction, discusses linguistic coverage issues, and explores the development of natural language widgets and their integration into multi user interfaces[1].The development of language technology has been facilitated by two technical breakthroughs: the first emphasizes empirical approaches and the second highlights networked machines [2].Natural language and databases are core components of information systems, and NLP techniques may substantially enhance most phases of query processing, natural language understanding and the information system [3-5].

By means of developed or used methods, metrics and measures, NLP has accelerated scientific advancement in human language such as machine translation[6-7], automated extraction systems from free-texts[8], the semantics-originated Generalized Upper Model of a linguistic ontology [9], artificial grammar learning (AGL) system[10], NIMFA[11], etc. Understanding natural language involves context-sensitive discrimination among word senses, and a growing awareness is created to develop an indexed domain-independent knowledge base that contains linguistic knowledge [12-17].

There are a lot of helpful NLP models for linguistic research focusing on various application areas, e.g. Zhou & Hripcsak' medical NLP model and Plant& Murrell's dialogue system.

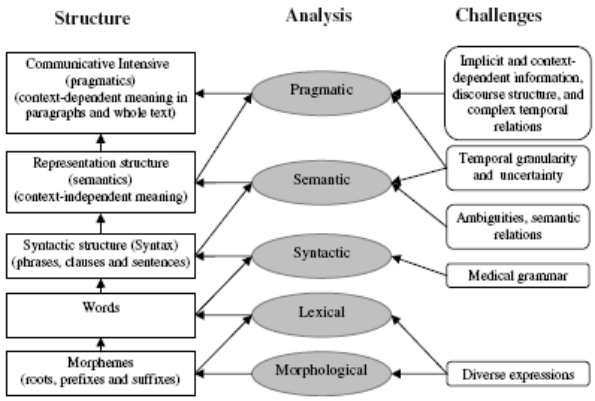

Figure 1 Zhou & Hripcsak' Medical NLP Model

Zhou & Hripcsak' medical NLP model comprises three parts, i.e. "structure", "analysis" and "challenges". "Analysis" consists in morphological, lexical, syntactic, semantic and pragmatic parts. Morphology and lexical analysis determine the sequences of morphemes used to create words. Syntax emphasizes the structure of phrases and sentences to combine multiple words.

Semantics highlights the formation of the meaning or interpretation of the words. Pragmatics concerns the situation of how context affects the interpretation of the sentences and of how sentences combine to form discourse. [18]

Plant& Murrell's Dialogue NLP System discusses the importance of Backus–Naur Form (BNF). This system analyzes the possibility for any user who understands formal grammars to replace or upgrade the system or to produce all possible parses of the input query without requiring any programming.

In the model, BNF is extended with simple semantic tags. The matching agent searches through a knowledge base of scripts and selects the most closely matching one. In this model, BNF is very helpful and useful for system to analyze natural language. [19]

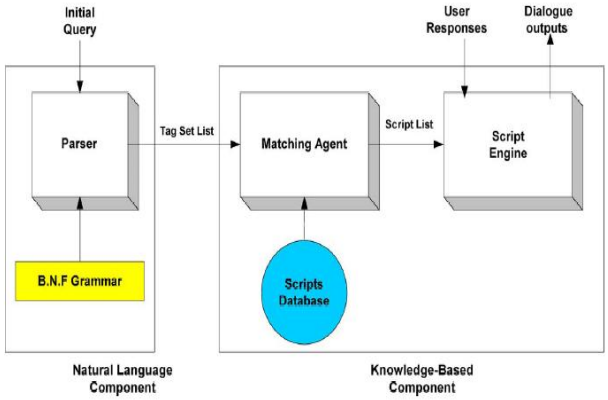

Figure 2 Plant& Murrell's Dialogue NLP System

The computational analysis of Garden Path (GP) sentences is one of the important branches of NLP for these sentences are hard for machine to translate if there is no linguistic knowledge to support.

GP sentences are grammatically correct and its interpretation consists of two procedures: the prototype understanding and the backtracking parsing. At the first time, readers most likely interpret GP sentences incorrectly by means of cognitive prototype. With the advancement of understanding, readers are lured into a parse that turns out to be a dead end. With the help of special word or phrase, they find that the syntactic structure which is being built up is different from the structure which has been created, namely it is a wrong path down which they have been led. Thus they have to return and reinterpret, which is called backtracking. "Garden path" here means "to be led down the garden path", meaning "to be misled". Originally, this phenomenon is analyzed by the psycholinguists to illustrate the fact that human beings process language one word at a time when reading. Now, GP phenomenon attracts a lot of interest of scholars from perspectives of syntax[20-24], semantics[25-28], pragmatics[29-30], psychology[31-34], computer and cognitive science[35-38].

In this paper, Context Free Grammar (CFG) and BNF will be used to discuss the automatic parsing of GP sentences. Meanwhile, the pre-grammatical sentences, common sentences and ambiguous sentences will be analyzed from the perspective of computational linguistics as the comparison and contrast to GP sentences.

II. The NLP-Based Analyses Of Non-GP Sentences

Non-GP sentences in this paper include the pre-grammatical sentences, common sentences and ambiguous sentences, all of which are shown how different they are from GP sentences.

A. Analysis of Pre-Grammatical Sentences

A pre-grammatical sentence is incorrect in grammar even though we can guess the meaning by the separated words or phrases. According to CFG, this kind of sentence fails to be parsed successfully.

Example 1: *The new singers the song.

G={Vn, Vt, S, P}

Vn={Det, Adj, N, NP, S, VP, V}

Vt={the, new, singers, song}

S=S

P:

1. S→NP VP
2. NP→Det N
3. NP→Det Adj N
4. VP→V NP
5. Det→{the}
6. N→{singers, song}
7. Adj→{new}
8. V→{?}

The new singers the song

Det new singers the song (5)

a. Det Adj singers the song (7)
b. Det Adj N the song (6)
c. NP the song (3)
d. NP Det song (5)
e. NP Det N (6)
f. NP NP (2)
g. FAIL

From the parsing of Example 1, we can see the whole structure of sentence is [The new singers]NP+[the song]NP，namely the absence of V is the reason why it fails to be parsed successfully.

In a pre-grammatical sentence, the syntactic structure is not correct and the relationships among the parts are isolated even though sometimes the possible meaning of the sentence can be inferred from the evidence. For example, in the programming rules of (8), we can enter a lot of related verbs to rewrite example 1, e.g. V→{hear/play/write/sing/ record}. Thus the pre-grammatical sentence can be created into a common one.

B. Analysis of Common Sentences

A common sentence is grammatically acceptable and both CFG and BNF can parse it smoothly and successfully. If "record(verb)" is added into example 1, the formed sentence is a common one.

Example 2：The new singers record the song.

G={Vn, Vt, S, P}

Vn={Det, Adj, N, NP, V, VP, S}

Vt={the, new, singers, record, song}

S=S

P:

1. S→NP VP
2. NP→Det N
3. NP→Det Adj N
4. VP→V NP
5. Det→{the}
6. N→{singers, song}
7. Adj→{new}
8. V→{record}

The new singers record the song

a.	Det new singers record the song	(5)
b.	Det Adj singers record the song	(7)
c.	Det Adj N record the song	(6)
d.	NP record the song	(3)
e.	NP V the song	(8)
f.	NP V Det song	(5)
g.	NP V Det N	(6)
h.	NP V NP	(2)
i.	NP VP	(4)
j.	S	(1)
k.	SUCCESS	

The syntactic structure of example 2 is [The new singers] NP+[record]V+[the song]NP, and the whole parsing is smooth.

Backus-Naur Form (BNF) is another useful formal language to describe the parsing of NLP. The details of BNF definition are as follows.

```
syntax ::=
    rule ::= identifier "::=" expression
    expression ::= term { "|" term }
    term ::= factor
    factor ::= identifier |
    quoted_symbol |
    "(" expression ")" |
    "[" expression "]" |
    "{" expression "}"
    identifier ::= letter { letter | digit }
quoted_symbol ::= """ """
```

Thus we can use BNF to define Augmented Transition Network (ATN) which will be introduced to analyze the related sentences in this paper.

<ATN>::=<State Arc>{<State Arc>}

<State Arc>::=<State><Arc>{<Arc>}

<Arc>::=CAT<Category><Preaction>

|PUSH<State>< Preaction >

|TST<Node>< Preaction >

|POP<Expression><Test>

<Preaction>::=<Test>{<Action>}<Terminal Action>

<Action>::=SETR<Register><Expression>

| SENDR<Register><Expression>

| LIFTR<Register><Expression>

<Terminal Action>::= TO<State>[<Form>]

| JUMP<State>[<Form>]

<Expression>::=GETR<Register>|*

| GETF<Feature>

|APPEND<Register><Expression>

|BUILD<Fragment>{<Register>}

In the semantic network, some nodes are associated with lexicon entries. In order to analyze example 2 clearly and concisely, we find a detailed description of lexicon is necessary besides the grammatical analysis. “CTGY” means category; “PRES”, present; “NUM”, number; “SING”, singular.

(The((CTGY. DET)))

(New((CTGY. ADJ)))

(Singers((CTGY.N) (NUM. PLURAL)))

(Record((CTGY.V)(PAST.RECORDED)(PASTP. RECORDED)))

(Record((CTGY. V) (TENSE.PRES))

(Song((CTGY. N) (NUM. SING)))

Based on the evidence discussed above, we can create an augmented transition network to analyze example 2.

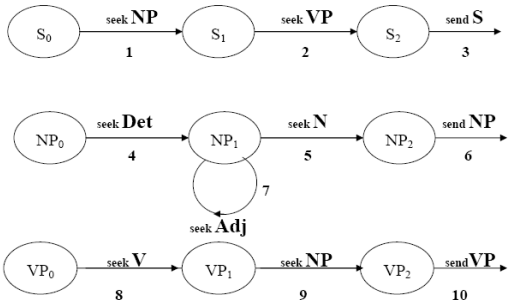

Figure 3 ATN of Example 2

The ATN in Fig. 3 shows the details of parsing of example 2, which belongs to the category of common sentence. There is no backtracking or ambiguity existing in the procedure shown below.

1. System tries to seek NP in arc 1 and then PUSH NP <The new singers> to NP subnet;

2. NP subnet begins to parse NP <The new singers>. In arc 4, Det <the> is set in register.

3. In arc 7, Adj <new> is analyzed and the result is set in register.

4. In arc 5, N<singers> is interpreted.

5. In arc 6, the result of parsing in NP subnet is popped to general net in arc 1.

6. Again in arc 1, the popped result is set in register.

7. In arc 2, system starts to seek VP<record the song> and PUSH to VP subnet.

8. VP subnet begins to parse VP<record the song>. In arc 8, V<record> is set in register.

9. In arc 9, VP subnet begins to interpret NP <the song>. There is no related rule to support the procedure in this VP subnet and as a result, the sub-sub-net of NP is activated again. NP <the song> is pushed to NP subnet.

10. NP sub-sub-net begins to parse NP<the song>. In arc 4, Det <the> is set in register again.

11. In arc 5, N <song> is parsed.

12. In arc 6, the result of parsing in NP sub-sub-net is popped to VP subnet.

13. In arc 9, NP<the song> is set in register.

14. In arc 10, VP<record the song> is popped.

15. In arc 2, the parsing result of VP subnet is set.

All the parsing results of subnets and sub-subnets show that S<the new singers record the song> is grammatically and semantically acceptable and reasonable. The information is set in register. System returns "SUCCESS" and parsing is over.

The algorithm of parsing discussed above can be found in Table 1, in which "Number" means the steps of parsing; "Complexity", the hierarchical levels of net; "Arc" or "A-?", the respective numbers shown in Fig. 3; "Programming", the BNF description.

C. Analysis of Ambiguous Sentences

An ambiguous sentence has more than one possible meaning, any of which can convey and carry the similar, different and even opposite information.

Example 3 : The detective hit the criminal with an umbrella.

The example above brings syntactic ambiguity for the different syntactic structures convey different meanings.

Number	Complexity	Arc	Programming
1	I	A-1	PUSH NP <The new singers>
2	II	A-4	SETR Det<The>
3		A-7	SETR Adj<new>
4		A-5	SETR N< singers >
5		A-6	POP NP
6	I	A-1	SETR NP<The new singers>
7		A-2	PUSH VP<record the song>
8	II	A-8	SETR V<record>
9		A-9	PUSH NP<the song>
10	III	A-4	SETR Det<the>
11		A-5	SETR N<song>
12		A-6	POP NP
13	II	A-9	SETR NP<the song>
14		A-10	POP VP
15	I	A-2	SETR VP<record the song>
16		A-3	SETR<The new singers record the song>
SUCCESS			

Table 1 Parsing Algorithm of Example 2

In example 3, two meanings are carried. The first is the detective using an umbrella hit the criminal, while the other is the detective hit the criminal who is carrying an umbrella.

G={Vn, Vt, S, P}

Vn={Det, N, NP, V, VP, S, Prep, PP}

Vt={the, detective, hit, criminal, with, an, umbrella}

S=S

P:

1. S→NP VP
2. NP→NP PP
3. NP→Det N
4. PP→Prep NP
5. VP→VP PP
6. VP→V NP
7. PP→Prep NP
8. Det→{the, an}
9. N→{detective, criminal, umbrella}
10. Prep→{with}
11. V→{hit}

(The((CTGY. DET)))

(Detective((CTGY.N) (NUM. SING)))

(Hit((CTGY. V) (PAST. HIT) (PASTP. HIT)))

(Hit((CTGY. V) (ROOT. HIT) (TENSE.PAST)))

(Hit((CTGY. V) (ROOT. HIT) (TENSE.PASTP)))

(Criminal ((CTGY.N) (NUM. SING)))

(With((CTGY.PREP)))

(An((CTGY. DET)))

(Umbrella((CTGY.N) (NUM. SING)))

The detective hit the criminal with an umbrella.

a. Det detective hit the criminal with an umbrella (8)
b. Det N hit the criminal with an umbrella (9)
c. NP hit the criminal with an umbrella (3)
d. NP V the criminal with an umbrella (11)
e. NP V Det criminal with an umbrella (8)
f. NP V Det N with an umbrella (9)
g. NP V NP with an umbrella (3)
h. NP VP with an umbrella (6)
i. NP VP Prep an umbrella (10)
j. NP VP Prep Det umbrella (8)
k. NP VP Prep Det N (9)
l. NP VP Prep NP (3)
m. NP VP PP (4)
n. NP VP (5)
o. S (1)
p. SUCCESS

Based on the parsing above, we can find the first exact meaning of example 3 is "The detective using an umbrella hit the criminal". Another parsing which means "The detective hit the criminal who is carrying an umbrella" is shown as follows.

The detective hit the criminal with an umbrella

a. Det detective hit the criminal with an umbrella (8)
b. Det N hit the criminal with an umbrella (9)
c. NP hit the criminal with an umbrella (3)
d. NP V the criminal with an umbrella (11)
e. NP V Det criminal with an umbrella (8)
f. NP V Det N with an umbrella (9)
g. NP V NP with an umbrella (3)
h. NP V NP Prep an umbrella (10)
i. NP V NP Prep Det umbrella (8)
j. NP V NP Prep Det N (9)
k. NP V NP Prep NP (3)
l. NP V NP PP (4)
m. NP V NP (2)
n. NP VP (6)
o. S (1)
p. SUCCESS

In ATN created by means of example 3, three subnets are involved, i.e. NP subnet, VP subnet and PP subnet. S net is the general net. The reason why the different meanings of example 3 can be expressed lies in the attached structures of PP subnet. When PP subnet is attached to VP subnet, namely VP→VP PP is activated, the parsing result is "The detective using an umbrella hit the criminal". When PP subnet serves NP subnet, i.e. NP → NP PP, the interpretation is "The detective hit the criminal who is carrying an umbrella".

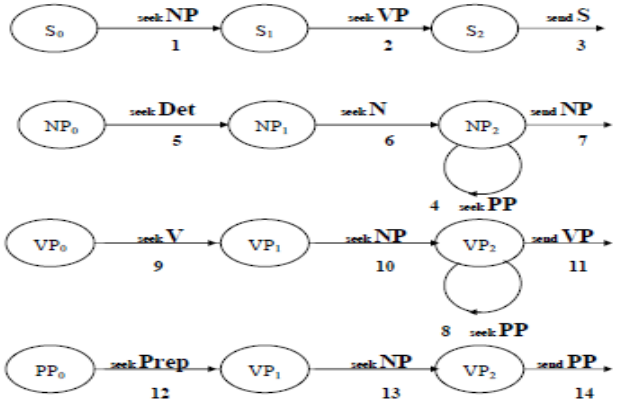

Figure 4 ATN of Example 3

From the Fig. 4, we can notice the difference of PP subnet which can be attached to NP subnet in arc 4 or to VP subnet in arc 8.

The parsing algorithm of example 3 in "VP→VP PP" includes 24 steps and highest level of syntactic structure is "IV".

1 In arc1, S-net seeks NP<the detective>. NP subnet used to parse noun phrase is activated.
2 In arc 5, NP subnet finds Det<the>.
3 In arc 6, N<detective> is parsed and set in register.
4 In arc 7, the parsing result is popped up to arc 1 where it is pushed.
5 In arc 1, NP<the detective> is set in register.
6 In arc 2, S-net seeks VP and the other part of VP<hit the criminal with an umbrella> is pushed down to VP subnet.
7 In arc 9, VP subnet seeks V<hit> firstly.
8 In arc 10, subnet seeks NP, and NP<the criminal> is pushed again to NP subnet to interpret.
9 In arc 5, NP subnet finds Det<the>.
10 In arc 6, NP subnet seeks N<criminal>.
11 In arc 7, the result of parsing of NP<the criminal> is popped up to arc 10 where it is pushed down.
12 In arc 10, NP<the criminal> is set in register.
13 In arc 8, VP subnet seeks PP<with an umbrella> and PP subnet is activated.

14 In arc 12, PP subnet finds Prep<with>.

15 In arc 13, PP subnet tries to parse NP <an umbrella> and for the third time, NP subnet is provided for the parsing.

16 In arc 5, NP subnet searches for Det<an>.

17 In arc 6, N<umbrella> is parsed in NP subnet.

18 In arc 7, the result is popped back to arc 13.

19 In arc 13, NP <an umbrella> is set in register.

20 In arc 14, PP<with an umbrella> is parsed successfully and it is popped up to arc 8.

21 In arc 8, PP<with an umbrella> is set in register.

22 In arc 11, the parsing of VP<hit the criminal with an umbrella> is finished and system has the result popped up to arc 2.

23 In arc 2, VP<hit the criminal with an umbrella> is set in register.

24 In arc 3, S<the detective hit the criminal with an umbrella> is parsed completely. System returns "SUCCESS" and parsing is over.

Number	Complexity	Arc	Programming
1	I	A-1	PUSH NP<The detective>
2	II	A-5	SETR Det<The>
3		A-6	SETR N<detective>
4		A-7	POP NP
5	I	A-1	SETR NP<The detective>
6		A-2	PUSH VP<hit the criminal with an umbrella>
7	II	A-9	SETR V<hit>
8		A-10	PUSH NP<the criminal>
9	III	A-5	SETR Det<the>
10		A-6	SETR N<criminal>
11		A-7	POP NP
12	II	A-10	SETR NP<the criminal>
13		A-8	PUSH PP <with an umbrella>
14	III	A-12	SETR Prep<with>
15		A-13	PUSH NP<an umbrella>
16	IV	A-5	SETR Det<an>
17		A-6	SETR N<umbrella>
18		A-7	POP NP
19	III	A-13	SETR NP <an umbrella>
20		A-14	POP PP <with an umbrella>
21	II	A-8	SETR PP <with an umbrella>
22		A-11	POP VP<hit the criminal with an umbrella>
23	I	A-2	SETR VP< hit the criminal with an umbrella >
24		A-3	SETR<The detective hit the criminal with an umbrella >
SUCCESS			

Table 2 Parsing Algorithm of Example 3 in "VP→VP PP"

The parsing algorithm of example 3 in "NP→NP PP" also has 24 steps and highest level of syntactic structure is "V", which means this parsing needs more cognitive or system burden to parse.

From Step 1 to Step 7, system parses example 3 along the same path in which both NP<the detective> and V<hit> are interpreted successfully without the existence of ambiguity. The same algorithm can be seen in both Table 2 and Table 3.

From Step 8, the difference appears. For the sake of clear and concise explanation, we start the algorithm used in Table 3 from step 8.

8 In arc 10, VP subnet seeks NP. Different from Step 8 in Table 2 where NP<the criminal> is pushed down to NP subnet, NP<the criminal with an umbrella> in Table 3 is pushed down, which means <with an umbrella> is just a modifier for <the criminal>.

9 In arc 5, NP subnet finds Det<the>.

10 In arc 6, NP subnet seeks N<criminal>.

11 In arc 4, NP subnet seeks PP<with an umbrella>, which will be pushed down to PP subnet.

12 In arc 12, PP subnet finds Prep<with>.

13 In arc 13, PP subnet seeks NP. NP<an umbrella> is pushed down to NP subnet again.

14 In arc 5, NP subnet seeks Det<an>.

15 In arc 6, N<umbrella> is parsed in NP subnet.

16 In arc 7, the result of parsing NP<an umbrella> is popped back to arc 13.

17 In arc 13, NP<an umbrella> is set in register.

18 In arc 14, the parsing result of PP<with an umbrella> is popped back to arc 4.

19 In arc 4, PP<with an umbrella> is set in register.

20 In arc 7, the parsing result of NP<the criminal with an umbrella> is popped back to arc 10.

21 In arc 10, NP<the criminal with an umbrella> is set in register.

22 In arc 11, the result of parsing VP<hit the criminal with an umbrella> is popped up to arc 2.

23 In arc 2, VP<hit the criminal with an umbrella> is set in register.

24 In arc 3, S<the detective hit the criminal with an umbrella> is parsed smoothly. System returns "SUCCESS" and parsing is over.

The difference between Table 2 and Table 3 shows that "VP→VP PP" parsing is easier than "NP→NP PP" parsing since the first is less complex than the second. This provides the evidence that there is a default parsing even though more than one interpretation is involved in an ambiguous sentence.

In example 3, "VP→VP PP" algorithm in which the sentence is parsed into "The detective using an umbrella hit the criminal" is the default interpretation.

Besides syntactic ambiguity shown in example 3, the existence of homographs is another important model to produce multi-meaning.

Number	Complexity	Arc	Programming
1	I	A-1	PUSH NP <The detective>
2	II	A-5	SETR Det<The>
3		A-6	SETR N<detective>
4		A-7	POP NP
5	I	A-1	SETR NP<The detective>
6		A-2	PUSH VP<hit the criminal with an umbrella>
7	II	A-9	SETR V<hit>
8		A-10	PUSH NP<the criminal with an umbrella >
9	III	A-5	SETR Det<the>
10		A-6	SETR N<criminal>
11		A-4	PUSH PP<with an umbrella >
12	IV	A-12	SETR Prep<with>
13		A-13	PUSH NP<an umbrella>
14	V	A-5	SETR Det<an>
15		A-6	SETR N<umbrella>
16		A-7	POP NP
17	IV	A-13	SETR NP<an umbrella>
18		A-14	POP PP<with an umbrella >
19	III	A-4	SETR PP<with an umbrella >
20		A-7	POP NP<the criminal with an umbrella >
21	II	A-10	SETR NP<the criminal with an umbrella >
22		A-11	POP VP <hit the criminal with an umbrella>>
23	I	A-2	SETR VP< hit the criminal with an umbrella >
24		A-3	SETR<The detective hit the criminal with an umbrella >
SUCCESS			

Table 3 Parsing Algorithm of Example 3 in "NP→NP PP"

Example 4: Failing student looked hard.

In example 4, both "failing" and "hard" have two meanings, namely, "failing(adj or Grd)" and "hard(adj or adv)". The semantic network of lexicon conveys the information.

(Failing((CTGY. GRD)))

(Failing((CTGY. ADJ)))

(Student((CTGY.N) (NUM. SING)))

(Look((CTGY.V)(PAST.LOOKED) (PASTP.LOOKED)))

(Looked((CTGY. V) (ROOT.LOOK) (TENSE. PAST)))

(Hard((CTGY. ADJ)))

(Hard((CTGY. ADV)))

From the lexicon, we can see the difference of homographs, which lead to four ambiguous sentences.

G={Vn, Vt, S, P}

Vn={N, NP, V, VP, S, Adv, Adj, Grd}

Vt={failing, student, looked, hard}

S=S

P:

1. S→NP VP
2. NP→Adj N
3. NP→Grd N
4. VP→V Adj
5. VP→V Adv
6. Adj→{failing, hard}
7. Grd→{failing}
8. N→{student}
9. V→{looked}
10. Adv→{hard}

Failing student looked hard (Grd+Adj)

a. Grd student looked hard (7)
b. Grd N looked hard (8)
c. NP looked hard (3)
d. NP V hard (9)
e. NP V Adj (6)
f. NP VP (4)
g. S (1)
h. SUCCESS

Failing student looked hard (Adj+Adj)

a. Adj student looked hard (6)
b. Adj N looked hard (8)
c. NP looked hard (2)
d. NP V hard (9)
e. NP V Adj (6)
f. NP VP (4)
g. S (1)
h. SUCCESS

Failing student looked hard (Grd+Adv)

a. Grd student looked hard (7)
b. Grd N looked hard (8)
c. NP looked hard (3)
d. NP V hard (9)
e. NP V Adv (10)
f. NP VP (5)
g. S (1)
h. SUCCESS

Failing student looked hard (Adj+Adv)

a. Adj student looked hard (6)
b. Adj N looked hard (8)
c. NP looked hard (2)

d. NP V hard (9)

e. NP V Adv (10)

f. NP VP (5)

g. S (1)

h. SUCCESS

According to the ambiguous interpretations of example 4, a special ATN used to analyze the sentence can be shown below.

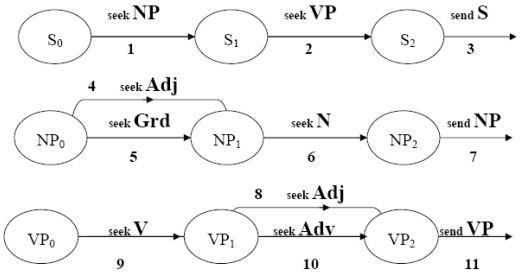

Figure 5 ATN of Example 4

In Fig. 5, we can see both NP subnet and VP subnet have bi-arcs which act as the same function of grammar. For example, arc 4 and arc 5 before NP1 exist in the same syntactic position and have the same function. Meanwhile, arc 8 and arc 10 before VP2 perform similar grammatical function in VP subnet. The BNF of example 4 is provided as follows.

Number	Complexity	Arc	Programming
1	I	A-1	PUSH NP <Failing student>
2	II	A-4	SETR Adj<Failing> A-5 SETR Grd<Failing>
3		A-6	SETR N<student>
4		A-7	POP NP
5	I	A-1	SETR NP<Failing student>
6		A-2	PUSH VP<looked hard>
7	II	A-9	SETR V<looked>
8		A-8	SETR Adj<hard> A-10 SETR Adv<hard>
9		A-11	POP VP
10	I	A-2	SETR VP<looked hard>
11		A-3	SETR<Failing student looked hard>
SUCCESS			

Table 4 Parsing Algorithm of Example 4

The whole BNF-based algorithm of example 4 is shown in Table 4, by which four interpretations discussed above can be parsed.

1 In arc 1, S-net needs NP and system pushes <failing student> to NP subnet.

2 In arc 4 and arc 5, NP subnet can parse <failing> as Adj or Grd. Both are correct and this is the first ambiguity. The parsing results are saved respectively.

3 In arc 6, N <student> is set in register.

4 In arc 7, NP<failing student> is parsed successfully (either Adj+N or Grd+N) and the result is popped back to arc 1 which needs the parsing result of NP<failing student>.

5 In arc 1, NP<failing student> is set in register

6 In arc 2, S-net seeks VP and <looked hard> is pushed down to the VP subnet.

7 In arc 9, V<looked> is found.

8 In arc 8 and arc 10, Adj<hard> or Adv<hard> is analyzed smoothly. This is the second ambiguity after the first one in arc 4 and arc 5.

9 In arc 11, the result of parsing (either V+Adj or V+Adv) is popped up to arc 2 where VP <looked hard> is pushed down.

10 In arc 2, the parsing result of VP <looked hard> is set in register.

11 In arc 3, S< failing student looked hard > is parsed successfully and smoothly, including four results of parsing, i.e. Adj+N+V+Adj, Adj+N+V+Adv, Grd+N+V+Adj, Grd+N+V+Adv. System returns "SUCCESS" and parsing is over.

From the discussion above, we can know a pre-grammatical sentence (e.g. example 1) is not good enough to meet the requirements of syntax for it fails to consist in the necessary components. A common sentence (e. g. example 2) is the essential part of natural language, and the exact expression is the core of the sentence. An ambiguous sentence comprises ambiguous structures (e.g. example 3) or ambiguous words (e.g. example 4), and any ambiguous interpretation is acceptable and understandable even though sometimes the parsing has different complexity.

III. The NLP-Focused Analyses Of GP Sentences

The parsing of a GP sentence includes two procedures, i.e. the prototype understanding and the backtracking parsing. The prototype understanding refers to the default parsing of cognition according to decoder's knowledge database. The backtracking parsing means the original processing breaks down and the decoder has to re-understand the GP sentence when the new information used to decode the sentence is provided linearly. Therefore, processing breakdown is the distinctive feature of the parsing of GP sentence.

Example 5: The opposite number about 5000.

The sentence is a GP one which contains the prototype understanding and backtracking parsing. The decoding experiences the breakdown of cognition.

G={Vn, Vt, S, P}

Vn={Det, Adj, N, LinkV, Adv, Num, NP, VP, S, NumP}

Vt={the, opposite, number, about, 5000}

S=S

P:

1. S→NP VP
2. NP→Det Adj
3. NP→Det Adj N
4. NumP→Adv Num

5. VP→LinkV NumP
6. Det→{the}
7. N→{number}
8. Adv→{about}
9. LinkV→{number}
10. Adj→{opposite}
11. Num→{5000}

(The((CTGY. DET)))

(Opposite((CTGY. ADJ)))

(Number((CTGY.LINKV)(PAST.NUMBERED)(PASTP. NUMBERED)))

(Number((CTGY. LINKV)(TENSE. PRES))

(Number((CTGY. N) (NUM. SING)))

(About((CTGY.ADV)))

(5000((CTGY. NUM)))

The opposite number about 5000

a. Det opposite number about 5000 (6)
b. Det Adj number about 5000 (10)
c. Det Adj N about 5000 (7)
d. NP about 5000 (3)
e. NP Adv 5000 (8)
f. NP Adv Num (11)
g. NP NumP (4)
h. FAIL and backtrack to another path:
i. Det Adj number about 5000 (10)
j. NP number about 5000 (2)
k. NP LinkV about 5000 (9)
l. NP LinkV Adv 5000 (8)
m. NP LinkV Adv Num (11)
n. NP LinkV NumP (4)
o. NP VP (5)
p. S (1)
q. SUCCESS

From the lexicon analysis of example 5, we can notice the significant difference between "number (noun)" and "number (linking verb)".

According to the interpretation in LDOCE, "number (noun)" can mean "a word or sign that represents an amount or a quantity" just in the sentence of "Five was her lucky number"; or "a set of numbers used to name or recognize someone or something" in the sentence of "He refused to swap it with opposite number Willie Carne after the game because he had promised it to the Mirror."

Besides the noun function, "number" can be parsed as "lingking verb". For example, in the sentence of "The men on strike now number 5% of the workforce", "number" is interpreted as "if people or things number a particular amount, that is how many there are."

Based on the discussion above, ATN of example 5 can be created.

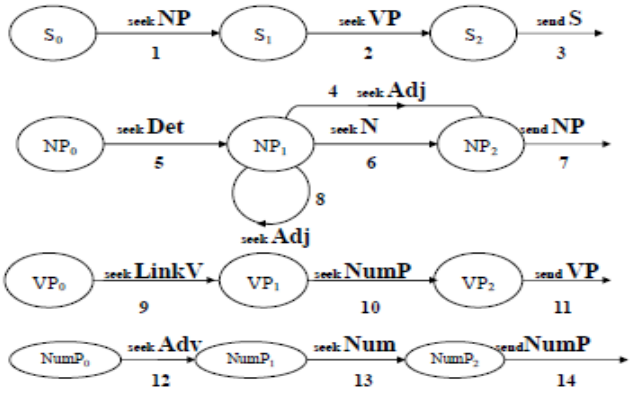

Figure 6 ATN of Example 5

In Fig. 6, the core of the parsing lies in NP subnet in which both "NP→Det Adj" and "NP→Det Adj N" are accepted. In cognitive system, "number (noun)" functions in order of priority while "number (lingking verb)"has a notably low probability. The difference of cognition can be shown in the ERP experiments and the psychological results develop the prototype ideas.[39-41]

The BNF-based algorithm of example 5 includes 22 steps during the parsing, which can be shown in Table 5.

1. In arc 1, S net firstly seeks NP. System pushes down to NP subnet. According to the cognitive knowledge of decoder, "number(noun)"in <the opposite number>" is firstly parsed.
2. In arc 5, Det<the> is set in register.
3. In arc 8, Adj<opposite> is interpreted successfully.
4. In arc 6, N<number> is set in register.
5. In arc 7, parsing result of NP<the opposite number> is popped up to arc 1 in S network where it is pushed down.
6. In arc 1, NP<the opposite number> is set in register.
7. In arc 2, S network seeks VP and tries to push down to VP subnet. But the left components<about 5000>fail to find V according to lexicon analysis. System returns "FAIL" and backtracks to the original path in arc 1 where another parsing can be chosen besides the original one. In example 5, the cognitive crossing lies in the difference of "number(noun)" and "number(linking verb)".
8. In arc 1, system seeks NP and <the opposite> instead of the original <the opposite number> is pushed down to NP subnet.

9. In arc 5, Det<the> is set in register.
10. In arc 4, Adj<opposite> is parsed.
11. In arc 7, NP<the opposite> is parsed successfully and sent back to arc 1.
12. In arc 1, the parsing result of NP<the opposite> is set in register.
13. In arc 2, VP<number about 5000> is pushed down to VP subnet.
14. In arc 9, <number> is interpreted as a linking verb according to (Number((CTGY. LINKV))), and the result of parsing is set in register.
15. In arc 10, VP subnet seeks NumP<about 5000>. NumP subnet is activated.
16. In arc 12, the interpretation of Adv<about> is set in register.
17. In arc 13, the number <5000> is parsed.
18. In arc 14, NumP<about 5000> is popped up to arc 10.
19. In arc 10, the result of parsing NumP<about 5000> is set in register.
20. In arc 11, after parsing VP<number about 5000> successfully and smoothly, system returns to arc 2.
21. In arc 2, VP<number about 5000> is set in register.
22. In arc 3, both NP<the opposite> and VP<number about 5000> are set in register and the whole parsing of S<The opposite number about 5000> is completed. System returns "SUCCESS" and parsing is over.

Number	Complexity	Arc	Programming
1	I	A-1	PUSH NP<The opposite number>
2	II	A-5	SETR Det<The>
3		A-8	SETR Adj<opposite>
4		A-6	SETR N<number>
5		A-7	POP NP
6	I	A-1	SETR NP<The opposite number>
7		A-2	PUSH VP<???>
Backtracking			
8	I	A-1	PUSH NP<The opposite>
9	II	A-5	SETR Det<the>
10		A-4	SETR Adj<opposite>
11		A-7	POP NP
12	I	A-1	SETR NP<The opposite>
13		A-2	PUSH VP<number about 5000>
14	II	A-9	SETR LinkV<number>
15		A-10	PUSH NumP<about 5000>
16	III	A-12	SETR Adv<about>
17		A-13	SETR Num<5000>
18		A-14	POP NumP
19	II	A-10	SETR NumP<about 5000>
20		A-11	POP VP
21	I	A-2	SETR VP<number about 5000>
22		A-3	SETR<The opposite number about 5000>
SUCCESS			

Table 5 Parsing Algorithm of Example 5

From the algorithm in Table 5, we can see the distinctive feature of parsing is the existence of "backtracking", at which breakdown happens and system has to return to the original crossing to find another road out. This optional procedure needs the help of lexical, semantic, grammatical and cognitive knowledge.

Example 6: The new record the song.

G={Vn, Vt, S, P}

Vn={Det, Adj, N, V, NP, VP, S}

Vt={the, new, record, song}

S=S

P:

1. S→NP VP
2. NP→Det Adj
3. NP→Det Adj N
4. NP→Det N
5. VP→V NP
6. Det→{the}
7. N→{record, song}
8. V→{record}
9. Adj→{new}

(The((CTGY. DET)))

(New((CTGY. ADJ)))

(Record((CTGY.V)(PAST.RECORDED)(PASTP. RECORDED)))

(Record((CTGY.V)(ROOT.RECORD)(TENSE. PRES)))

(Record((CTGY. N) (NUM. SING)))

(Song((CTGY.N) (NUM. SING)))

The new record the song

a. Det new record the song (6)
b. Det Adj record the song (9)
c. Det Adj N the song (7)
d. NP the song (3)
e. NP Det song (6)
f. NP Det N (7)
g. NP NP (4)
h. FAIL and backtrack to another path:
i. Det Adj record the song (9)
j. NP record the song (2)
k. NP V the song (8)
l. NP V Det song (6)
m. NP V Det N (7)
n. NP V NP (4)
o. NP VP (5)

p. S (1)

q. SUCCESS

From the parsing above, we can know example 6 is another GP sentence since there is breakdown in the processing. In example 6, "record(verb)" and "record(noun)" can be chosen randomly. However, NP<the new record> has a high probability of parsing. This is the reason why the priority parsing selects "record(noun)" rather than "record(verb)". The process of choosing can be shown in ATN networks.

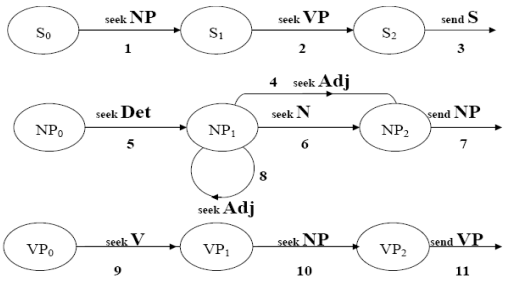

Figure 7 ATN of Example 6

In Fig. 7, NP subnet structure is the obvious reason why the GP phenomenon appears. Both NP→Det Adj and NP→ Det Adj N are reasonable and acceptable when "the new record" is parsed. Generally speaking, Adj is used to modify the Noun, the model of NP→Det Adj N is the prototype of parsing, and system interprets example 6 by means of this programming rule rather than NP→Det Adj. After completing the NP subnet parsing of <the new record>, system returns to S network to seek VP. However, the left phrase <the song> has no VP factor according to the lexicon knowledge, and system stops, backtracks and transfers to another programming rule, i.e. NP→Det Adj. Cognitive breakdown happens. The whole processing algorithm of example 6 is shown in Table 6.

1. In arc 1, S network needs NP information. The prototype of NP<the new record> has higher probability than NP<the new> in decoder's cognition, and NP<the new record> is pushed down to NP subnet.
2. In arc 5, system finds Det<the>.
3. In arc 8, Adj<new> is found.
4. In arc 6, N<record> is matched.
5. In arc 7, system finishes the parsing NP<the new record> and returns to arc 1.
6. In arc 1, the parsing result of NP<the new record> is saved.
7. In arc 2, system seeks VP information. However, no related lexicon knowledge is provided in (The((CTGY. DET))) and (Song((CTGY.N) (NUM. SING))). System fails and backtracks to arc 1 to find another programming rule of NP→Det Adj instead of NP→Det Adj N.
8. In arc 1, NP<the new> is chosen as a new alternative. NP subnet is activated once more.
9. In arc 5, Det<the> is set in register.

Number	Complexity	Arc	Programming
1	I	A-1	PUSH NP <The new record>
2	II	A-5	SETR Det<The>
3		A-8	SETR Adj<new>
4		A-6	SETR N<record>
5		A-7	POP NP
6	I	A-1	SETR NP<The new record>
7		A-2	PUSH VP<???>
Backtracking			
8	I	A-1	PUSH NP <The new>
9	II	A-5	SETR Det<the>
10		A-4	SETR Adj<new>
11		A-7	POP NP
12	I	A-1	SETR NP<The new>
13		A-2	PUSH VP<record the song>
14	II	A-9	SETR V<record>
15		A-10	PUSH NP <the song>
16	III	A-5	SETR Det<the>
17		A-6	SETR N<song>
18		A-7	POP NP
19	II	A-10	SETR NP <the song>
20		A-11	POP VP
21	I	A-2	SETR VP<record the song>
22		A-3	SETR<The new record the song>
SUCCESS			

Table 6 Parsing Algorithm of Example 6

10. In arc 4, Adj<new> is interpreted successfully.
11. In arc 7, NP<the new> is parsed completely and the result is popped back to arc 1.
12. In arc 1, the popped result of NP is set in register.
13. In arc 2, system seeks VP and <record the song> is pushed down to VP subnet.
14. In arc 9, VP subnet is activated and the knowledge of (Record((CTGY.V)(ROOT.RECORD)(TENSE. PRES))) helps system regard <record> as verb.
15. In arc 10, VP subnet seeks NP. The NP<the song> is pushed down to NP subnet.
16. In arc 5, NP subnet is activated again. Det<the> is set in register.
17. In arc 6, N<song> is set in register.
18. In arc 7, NP<the song> is parsed completely and the result is popped up to arc 10 where it is pushed down.
19. In arc 10, the parsing result of NP<the song> is set in register.
20. In arc 11, VP<record the song> is parsed successfully and popped up to arc 2.
21. In arc 2, the parsing result of VP<record the song> is set in register.

22. In arc 3, system finishes the parsing of NP<the new> and VP<record the song>. S<the new record the song> is saved. System returns "SUCCESS" and parsing is over.

From the discussion about example 5 and example 6, we can find both of them have the distinctive feature of "backtracking". The fact that high probability parsing in GP sentences has to be replaced by the low probability interpretation is the fundamental distinction from pre-grammatical sentences, common sentences and ambiguous sentences. Processing breaks down when system backtracks to find new path out.

Based on the analyses of computational linguistics shown above, we can see more likeness and unlikeness exist between the ambiguous sentences and GP sentences. An effective and systematic attempt at comparison and contrast may contribute to our understanding of the special phenomenon.

IV. The Comparison And Contrast Of Ambiguous Sentences And Gp Sentences

Ambiguous sentences and GP sentences have close similarities and significant differences in many aspects, e.g. lexicon knowledge, syntactic structures and decoding procedures.

A. The Similarity and Difference in Lexicon Knowledge

The lexicon knowledge is the basic information for system to parse and a detailed analysis of related category is essential and necessary. Let's firstly compare the similarity and contrast the difference among example 3, example 4, example 5 and example 6, which are shown as follows.

In example 3, the lexicon analysis includes Det<the, an>, N<detective, criminal>, Prep<with> and V<hit>. Since the singular noun N<detective> needs present verb <hits> or past verb <hit> to cooperate, example 3 must be a past tense rather than a present tense for there is no <hits> provided in the sentence. Example 3 is a structure-based ambiguous sentence and lexicon knowledge helps few for reducing ambiguities.

(The((CTGY. DET)))

(Detective((CTGY.N) (NUM. SING)))

(Hit((CTGY. V) (PAST. HIT) (PASTP. HIT)))

(Hit((CTGY. V) (ROOT. HIT) (TENSE.PAST)))

(Hit((CTGY. V) (ROOT. HIT) (TENSE.PASTP)))

(Criminal ((CTGY.N) (NUM. SING)))

(With((CTGY.PREP)))

(An((CTGY. DET)))

(Umbrella((CTGY.N) (NUM. SING)))

In example 4, lexicon knowledge contains the analyses of Grd<failing>, Adj<failing, hard>, N<student>, V<looked>, Adv<hard>. The homonyms of <failing> and <hard> bring the double ambiguities in the sentence, which results in four different meanings. The whole ambiguity lies in the lexical multi-meaning. Therefore, example 4 is the model of lexical ambiguity.

(Failing((CTGY. GRD)))

(Failing((CTGY. ADJ)))

(Student((CTGY.N) (NUM. SING)))

(Look ((CTGY.V)(PAST.LOOKED) (PASTP.LOOKED)))

(Looked((CTGY. V) (ROOT.LOOK) (TENSE. PAST)))

(Hard((CTGY. ADJ)))

(Hard((CTGY. ADV)))

In example 5, the lexical database comprises Det<the>, Adj<opposite>, LinkV<number>, N<number>, Adv<about>, and Number<5000>. The homonym <number> has two grammatical functions, i.e. linking verb and noun.

The different choices result in different sentences. According to the probability, NP<the opposite number> is the prototype parsing, and correspondingly, N<number> is adopted firstly even though this path is considered to be a dead end finally. Generally speaking, the lexical crossing leads to the processing breakdown of GP sentence.

(The((CTGY. DET)))

(Opposite((CTGY. ADJ)))

(Number((CTGY.LINKV)(PAST.NUMBERED)(PASTP. NUMBERED)))

(Number((CTGY. LINKV)(TENSE. PRES))

(Number((CTGY. N) (NUM. SING)))

(About((CTGY.ADV)))

(5000((CTGY. NUM)))

In example 6, a lot of lexicons are analyzed, i.e. Det<the>, Adj<new>, V<record> and N<record, song>. The meaning of <record> diverges markedly when N<record> is replaced by V<record> to meet the requirements of syntax. This sentence is another example in which processing breakdown is a direct consequence of lexical divergence.

(The((CTGY. DET)))

(New((CTGY. ADJ)))

(Record((CTGY.V)(PAST.RECORDED)(PASTP. RECORDED)))

(Record((CTGY.V)(ROOT.RECORD)(TENSE. PRES)))

(Record((CTGY. N) (NUM. SING)))

(Song((CTGY.N) (NUM. SING)))

From the discussion above, we can see the existence of homonyms is an obvious reason which brings ambiguous phenomenon and GP effect, just as in example 4, example 5 and example 6.

However, this is not the only reason for the appearance of ambiguity or GP phenomenon. Sometimes, the divergence of syntactic structures also leads to ambiguity or GP effect.

B. The Similarity and Difference in Syntactic Structures

Stanford parser is a very useful parser which is created by means of both highly optimized PCFG (probabilistic context free grammar), lexicalized dependency parsers and lexicalized PCFG. "Probabilistic parsers use knowledge of language gained from hand-parsed sentences to try to produce the most likely analysis of new sentences." The Stanford parser can be used to parse example3, example 4, example 5 and example 6 on line. The results of syntactic structures are provided as follows.

In example 3, the tags include <the/DT>, <detective/NN>, <hit/VBD>, <criminal/NN>, <with/IN>, <an/DT>, and <umbrella/NN>. The parsing structure is a full parsed one in which <the detective> is parsed as NP; <hit the criminal with an umbrella>, VP; <the criminal>, sub-net's NP; <with an umbrella>, sub-net's PP parsed as a modifier for <hit>; <with>, sub-net's PP; <an umbrella>, sub-sub-net's NP. The hierarchical structure is similar to the complexity in Table 2. Stanford parser provides one of the two interpretations, namely model of "VP→VP PP" rather than the model of "NP →NP PP" since the former has higher probability than the latter from the perspective of statistics. In other words, "VP→ VP PP" is the prototype parsing for its simpler syntactic structure.

(ROOT
 (S (I)
  (NP (DT the) (NN detective)) (II)
  (VP (VBD hit) (II)
   (NP (DT the) (NN criminal)) (III)
   (PP (IN with) (III)
    (NP (DT an) (NN umbrella)))) (IV)
 (. .)))

In example 4, the tags are <Failing/NN>, <student/NN>, <looked/VBD> and <hard/JJ>. This is another whole parsed structure in which all the components are interpreted successfully. The word of <failing> is considered Noun (i.e. Grd); <hard>, JJ (i.e. Adj). The parsed syntactic structure is similar to "Grd+Adj" which is the highest probability in statistics of parsing database among four ambiguous models. The hierarchical level is II shown in Table 4.

(ROOT
 (S (I)
  (NP (NN Failing) (NN student)) (II)
  (VP (VBD looked)(ADJP (JJ hard))) (II)
 (. .)))

In example 5, tags are < the/DT >, < opposite/JJ >, <number/NN >, < about/RB> and <5000/CD >. According to Stanford parser, this is a part-parsed sentence since the final result is NP rather than S, which shows the prototype of NP<the opposite number> has the higher probability than NP<the opposite>. In other words, Stanford parser only finishes the first part of the parsing before the backtracking in Table 5.

(ROOT
 (NP (I)
  (NP (DT the) (JJ opposite) (NN number)) (II)
  (QP (RB about) (CD 5000)) (II)
 (. .)))

In example 6, tags comprise <the/DT>, <new/JJ>, <record/NN >, and <song/NN>. This is another example of part-parsed structure in which only the programming rule of N→{record} is adopted while V→{record} fails to be used. That means NP<the new record> has stronger statistical probability than NP<the new >. Stanford parser only parses the steps from 1-7 in Table 6 and then system gives the final result is NP instead of S, which ignores the left parsing steps after the backtracking.

(ROOT
 (NP (I)
  (NP (DT the) (JJ new) (NN record)) (II)
  (NP (DT the) (NN song)) (II)
 (. .)))

From the discussion about syntactic structures, we can see both ambiguous sentences and GP sentences can have more than one syntactic structure. According to PCFG, the strongest probability parsing is the final result in Stanford parser. If another more complex structure is adopted, cognitive burden of decoders will be lifted and increased. Once this happens, another ambiguous sentence will be provided by means of the ambiguous syntactic structure besides the original one. On the contrast, if probability-based parsing returns the final result of a GP sentence as a part-parsed structure, the rule-based programming will be activated and a full-parsed new structure can be obtained only if the processing breakdown can be overcome.

During the re-parsing procedures, an ambiguous structure can bring different full-parsed results, while a GP sentence breaks down firstly for its part-parsed structure and then moves on to another full-parsed path. An ambiguous structure leads to multi-results, all of which are reasonable and acceptable while a GP sentence structure only brings one full-interpreted result besides the processing breakdown.

V. CONCLUSION

By comparing programming procedures, lexicon knowledge, parsing algorithms and syntactic structures between pre-grammatical sentences, common sentences, ambiguous sentences and GP sentences, we conclude that the formal methods of computational linguistics, e.g. CFG, BNF, and ATN, are useful for computational parsing. Pre-grammatical sentences have part-parsed structure and system returns the final result to be Phrases rather than S. Common sentences are normal in grammar and semantics, and there is

no lexical or syntactic crossing for parsing. Ambiguous sentences have ambiguity created by ambiguous structures or lexicons, both of which can bring full-parsed results. GP sentences comprise part-parsed structure built by the high statistical probability method, and full-parsed structures created by rule-based method. When the parsing shifts from part-parsed structure to the full-parsed one, processing breakdown of GP sentences occurs. This paper supports the idea raised by Pritchett [42]that processing breakdown is a distinctive feature in the parsing of a GP sentence.

ACKNOWLEDGMENT

This research is supported in part by grants of YB115-29 and YB115-41 from "the Eleventh Five-year" research projects of Chinese language application, and grant 11YJA740111 from the Ministry of education and science planning project.

REFERENCES

[1] B. Manaris, "Natural language processing: A human-computer interaction perspective," Advances in Computers, vol. 47, 1998, pp. 1-66.

[2] P. Jackson and F. Schilder, "Natural language processing: Overview," Encyclopedia of Language & Linguistics, 2006, pp. 503-518.

[3] D. E. Suranjan, P. A. N. Shuh-Shen, and A. B. Whinston, "Natural language query processing in a temporal database," Data & Knowledge Engineering, vol. 1, June 1985, pp. 3-15.

[4] E. Métais, "Enhancing information systems management with natural language processing techniques," Data & Knowledge Engineering, vol. 41, June 2002, pp. 247-272.

[5] G. Neumann, "Interleaving natural language parsing and generation through uniform processing," Artificial Intelligence, vol. 99, Feb. 1998, pp. 121-163.

[6] M. Maybury, "Natural language processing: System evaluation," Encyclopedia of Language & Linguistics, 2006, pp. 518-523.

[7] C. Mellish and X. Sun, "The semantic web as a linguistic resource: Opportunities for natural language generation," Knowledge-Based Systems, vol. 19, Sep. 2006, pp. 298-303.

[8] W. W. Chapman et al, "Classifying free-text triage chief complaints into syndromic categories with natural language processing," Artificial Intelligence in Medicine, vol. 33, Jan. 2005, pp. 31-40.

[9] J. A. Bateman, J. Hois, R. Ross, and T. Tenbrink, "A linguistic ontology of space for natural language processing," Artificial Intelligence, vol. 174, Sep. 2010, pp. 1027-1071.

[10] P. F. Dominey, T. Inui, and M. Hoen, "Neural network processing of natural language: Towards a unified model of corticostriatal function in learning sentence comprehension and non-linguistic sequencing," Brain and Language, vol. 109, June 2009, pp. 80-92.

[11] M. Stanojević, N. Tomašević, and S. Vraneš, "NIMFA – natural language implicit meaning formalization and abstraction," Expert Systems with Applications, vol. 37, Dec. 2010, pp. 8172-8187.

[12] V. R. Dasigi, and J. A. Reggia, "Parsimonious covering as a method for natural language interfaces to expert systems," Artificial Intelligence in Medicine, vol. 1, 1989, pp.49-60.

[13] S. J. Conlon, J. R. Conlon, and T. L. James, "The economics of natural language interfaces: Natural language processing technology as a scarce resource," Decision Support Systems, vol. 38, Oct. 2004, pp. 141-159.

[14] S. Menchetti, F. Costa, P. Frasconi, and M. Pontil, "Wide coverage natural language processing using kernel methods and neural networks for structured data," Pattern Recognition Letters, vol. 26, Sep. 2005, pp. 1896-1906.

[15] E. Alba, G. Luque, and L. Araujo, "Natural language tagging with genetic algorithms," Information Processing Letters, vol. 100, Dec. 2006, pp. 173-182.

[16] W. M. Wang, C. F. Cheung, W. B. Lee, and S. K. Kwok, "Mining knowledge from natural language texts using fuzzy associated concept mapping," Information Processing & Management, vol. 44, September 2008, pp. 1707-1719.

[17] P. Cimiano, P. Haase, J. Heizmann, M. Mantel, and R. Studer, "Towards portable natural language interfaces to knowledge bases – The case of the ORAKEL system," Data & Knowledge Engineering, vol. 65, May 2008, pp. 325-354.

[18] L. Zhou and G. Hripcsak, "Temporal reasoning with medical data—A review with emphasis on medical natural language processing," Journal of Biomedical Informatics, vol. 40, April 2007, pp. 183-202.

[19] R. Plant and S. Murrell, "A natural language help system shell through functional programming," Knowledge-Based Systems, vol. 18, Feb. 2005, pp. 19-35.

[20] J. L. Du, P. F. Yu, "Syntax-directed machine translation of natural language: Effect of garden path phenomenon on sentence structure," International Conference on Intelligent Systems Design and Engineering Applications, 2010, pp. 535－539.

[21] J. Häussler and M. Bader, "The assembly and disassembly of determiner phrases: Minimality is needed, but not sufficient," Lingua, 119(10), 2009, pp.1560-1580.

[22] T. Malsburg and S. Vasishth, "What is the scanpath signature of syntactic reanalysis?" Journal of Memory and Language, 65(2), 2011, pp.109-127.

[23] K. R. Christensen, "Syntactic reconstruction and reanalysis, semantic dead ends, and prefrontal cortex," Brain and Cognition, 73(1), 2010, pp. 41-50.

[24] T. G. Bever, "The cognitive basis for linguistic structures," In J. R. Hayes, Ed. Cognition and the Development of Language, New York: John Wiley and Sons, 1987, pp. 279-352.

[25] Y. H. Jin, "Semantic analysis of Chinese garden-path sentences," Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing, 2006, (7), pp. 33–39.

[26] K. Christianson et al, "Thematic roles assigned along the garden path linger," Cognitive Psychology, 2001, (42), pp. 368–407.

[27] M. P. Wilson and S. M. Garnsey, "Making simple sentences hard: Verb bias effects in simple direct object sentences," Journal of Memory and Language, 2009, 60(3), pp. 368-392.

[28] A. D. Endress and M. D. Hauser, "The influence of type and token frequency on the acquisition of affixation patterns: Implications for language processing," Journal of Experimental Psychology: Learning, Memory, and Cognition, vol. 37, Jan. 2011, pp. 77-95.

[29] D. J. Foss and C. M. Jenkins, "Some effects of context on the comprehension of ambiguous sentences," Journal of Verbal Learning and Verbal Behavior, 1973, (12), pp. 577.

[30] K. G. D. Bailey and F. Ferreira, "Disfluencies affect the parsing of garden-path sentences," Journal of Memory and Language, 2003, (49), pp. 183–200.

[31] M. Bader and J. Haussler, "Resolving number ambiguities during language comprehension," Journal of Memory and Language, 2009, (08).

[32] N. D. Patson et al, "Lingering misinterpretations in garden-path sentences: Evidence from a paraphrasing task," Journal of Experimental Psychology: Learning, Memory, and Cognition, 2009, 35(1), pp. 280-285.

[33] J. Feeney, D. Coley, and A. Crisp, "The relevance framework for category-based induction: Evidence from garden-path arguments," Journal of Experimental Psychology: Learning, Memory, and Cognition, vol. 36, July 2010, pp. 906-919.

[34] Y. Choi and J. C. Trueswell, "Children's (in)ability to recover from garden paths in a verb-final language: Evidence for developing control in sentence processing," Journal of Experimental Child Psychology, 2010, 106(1), pp. 41-61.

[35] P. F. Yu and J. L. Du, "Automatic analysis of textual garden path phenomenon: A computational perspective," Journal of Communication and Computer, 2008, 5 (10), pp. 58-65.

[36] B, McMurray, M. K. Tanenhaus and R. N. Aslin, "Within-category VOT affects recovery from 'lexical' garden-paths: Evidence against phoneme-level inhibition," Journal of Memory and Language, 2009, 60(1), pp. 65-

91.

[37] P. L. O'Rourke and C. V. Petten, "Morphological agreement at a distance: Dissociation between early and late components of the event-related brain potential," Brain Research, 2011, 1392(5), pp. 62-79.

[38] J. L. Du and P. F. Yu, "Towards an algorithm-based intelligent tutoring system: computing methods in syntactic management of garden path phenomenon," Intelligent Computing and Intelligent Systems, 2010, (10), pp. 521-525.

[39] E. Malaia, R. B. Wilbur, and C. Weber-Fox, "ERP evidence for telicity effects on syntactic processing in garden-path sentences," Brain and Language, 2009, 108(3), pp.145-158.

[40] A. Staub, "Eye movements and processing difficulty in object relative clauses," Cognition, 2010, 116(1), pp. 71-86.

[41] M. O. Ujunju, G. Wanyembi and F. Wabwoba. "Evaluating the role of information and communication technology (ICT) support towards processes of management in institutions of higher learning," International Journal of Advanced Computer Science and Applications, 2012, 3(7), pp. 55-58.

[42] B. L. Pritchett, "Garden path phenomena and the grammatical basis of language processing," Language, 1988, (64), pp. 539-576.