June 15, 2022

SD-213

Cognitive Approach to Natural Language Processing

Micro-study

teaching.dessalles.fr/CANLP

Name: Kaixuan Zhu, Xinhao Li

More Sophisticated Grammar for Chart Parser

I. Abstract

In this micro-project, we extend the grammar by adding the feature structures to detect some basic grammar mistakes and meaning errors. Then, by modeling and unification, the Chart Parser can now parse legal sentences with conjunctions and relative clauses. The result is quite satisfying, but not so elegant in our point of view.

II. Problem

In this project, we want to solve three main problems: basic grammar check, conjunction words and relative clauses.

For basic grammar check, we hope that the Chart Parser can parse and check these sentences below:

a. the dog barks
b. the dogs bark
c. the dog bark **(Wrong and could not be parsed)**

That is to say, the Chart Parser can make the agreement between subject and verb.

For the conjunction part, the Chart Parser can now parse the sentences with conjunctions following the agreement between subject and verb. The detailed grammar rules will be explained later. For example, we can now parse these sentences below:

a. the dog and the neighbour grumble.
b. the dog or the neighbour grumbles.

However, we also focus on the sequence of personal pronouns, such as you, I, we,and they. For example,the sentence “You and I talk” is correct but “I and You talk” is incorrect. The sentence “We and you talk” is correct but the sentence “You and we talk” is incorrect.

For the relative clauses part, the goal is that not only can the Chart Parser parse the clauses in the sentence, but also it can recognize the agreement between subject and verb in the whole sentence. For example, we can now parse these sentences below:

a. I hate the **dog** that barks **(attributive clause)**
b. who I talk to about my sister **hates** the neighbour **(subject clause)**
c. I like who my sister gives the dog to **(object clause)**

but not:

a. I hate the **dogs** that barks **(attributive clause)**
b. who I talk to about my sister **hate** the neighbour **(subject clause)**

III. Method

In this part, we will describe the method we are applying to solve the problems above. At first, Kaixuan did the ***Basic Grammar Check***. And I did the parts of ***Conjunction*** and ***Relative Clauses***. Then, we merged these two functions together.

A. Basic grammar check

In this part, the first task is to add some grammar to the DCG for the agreement between subject and verb. In other words, the verb should change when the subject is third-person singular. In order to achieve this, we add some feature structure:

For noun phrases **np**, we mainly distinguish in three aspects: if the noun is singular or plural; if the noun is first-person, second-person or third-person; and if the noun is humain or not. We can see the following examples in detail:

```
n([number:1,person:1,is_person:1])    -->    [i].%    the    noun    i    is
singular,first-person and related to humain.
n([number:2,person:3,is_person:1]) --> [daughters].% the noun daughters are
plural,third-person and related to humain.
n([number:2,person:3,is_person:0])   -->   [dogs].   %   the   noun   dogs   are
plural,third-person but not related to humain.
```

For verb phrases **vp**, apart from adding the feature structures for the agreement between subject and verb, we also distinguish in the transitivity of a verb. We can look at the following examples for further comprehension:

```
v([subj:[number:1,person:3,is_person:_],transitive:0]) --> [grumbles].
  %the  intransitive  verb  grumble  corresponds  to  subject  third-person
singular.
v([subj:[number:2, person:1,is_person:_],transitive:1]) --> [like].
```

```
  %the transitive verb like corresponds to the subject first-person plural.
v([subj:[number:2, person:3,is_person:0],transitive:0]) --> [bark].
   %the intransitive verb correspond to the subject third-person plural
which could not be a person,like dogs.
```

B. Conjunction

In this part, we will try to add some grammar to the DCG so that the chart parser can recognize the sentence with conjunctions. To simplify the modeling, the conjunction words will be limited to **and**, **or** and **but**.

There are two types of conjunctions in our grammar, the conjunction between noun phrases (**np**) and the conjunction between verb phrases (**vp**) and sentences (**s**). The first type includes **and** and **or**. The latter contains **and**, **or** and **but**. The definition of these words are as follows:

```
conj_and --> [and]. %% conjunction for people
conj_or --> [or].

conj --> [and].     %% conjunction for verbs/sentences
conj --> [or].
conj --> [but].
```

Since the conjunction can only connect the phrases of the same type, the grammar is also not complex. We just need to add the connection of different phrases.

```
s --> s, conj, s.          %Two simple sentences connected with a
                           conjunction.

np --> np, conj_and, np. %Two simple noun phrases connected with "and".
np --> np, conj_or, np.  %Two simple noun phrases connected with "or".

vp --> vp, conj, vp.       %Two simple verb phrases connected with a
                           conjunction

pp --> pp, conj, pp.       %Two simple prepositional phrases connected
                           with a conjunction
```

The difficulty is mainly focused on how to unify the ***FS***. There is nothing to do with the 1st and the 4th predicates as **s** and **pp** don't have the ***FS***. The unification of **vp** is also simple as the verb phrases are connected to the same noun phrase. We just need to maintain the same **subj:*FS*** in the ***FS*** for **vp**.

When dealing with **np**, there are two problems to solve. The first is the order of personal pronouns. In English grammar, the order between different personal pronouns: we have to say "you, he and I" and "we, you and they". As the Chart Parser we are using doesn't support the predicates in curly brackets, we have to enumerate every possibility. The second is the unification of **number**, **person** and **is_person** in ***FS***. For **and**, the **number** will become plural (**2**) no matter how the two **np** are defined; the term **person** is personally defined; and

is_person is defined by the operation AND (**max(*Ip1*,*Ip2*)**) between **is_person** of two **np**. For **or**, the **number** will be defined by the maximum of two **number** in the two **np**; the term **person** is also personally defined; and **is_person** is defined by the operation OR (**Ip1*Ip2**) between the **is_person** of two **np**.

```
np([number:1,person:2,is_person:1])-->np([number:1,person:2,is_person:1]),
conj_or, np([number:_, person:_,is_person:_]). %you or ...
np([number:max(Num,1),person:1,is_person:1])-->np([number:Num,person:_,is_p
erson:_]), conj_or, np([number:1, person:1,is_person:1]). %... or i
np([number:max(Num1,Num2),person:3,is_person:max(Ip1,Ip2)])-->np([number:Nu
m1,person:3,is_person:Ip1]),conj_or,np([number:Num2,person:3,is_person:Ip2]
). %Other situations

np([number:2,person:2,is_person:1])-->np([number:1, person:2,is_person:1]),
conj_and, np([number:_, person:_,is_person:_]). %you and ...
np([number:2,person:1,is_person:1])-->np([number:_, person:_,is_person:_]),
conj_and, np([number:1, person:1,is_person:1]). %... and i
np([number:2,person:3,is_person:Ip1*Ip2])-->np([number:_,person:3,is_person
:Ip1]), conj_and, np([number:_, person:3,is_person:Ip2]). %Other situation

np([number:2,person:1,is_person:_])-->np([number:2, person:1,is_person:1]),
conj_or, np([number:_, person:_,is_person:_]). %we or ...
np([number:2,person:Person1,is_person:1])-->np([number:_,person:Person1,is_
person:1]), conj_or, np([number:2, person:3,is_person:_]). % ... or they

np([number:2,person:1,is_person:_])-->np([number:2, person:1,is_person:1]),
conj_and, np([number:2, person:_,is_person:_]). %we and ...
np([number:2,person:Person1,is_person:1])-->np([number:2,person:Person1,is_
person:1]), conj_and, np([number:2, person:3,is_person:_]). %... and they
```

C. Relative clauses

The relative clauses in our micro-project are restricted to attributive clauses and object (subject) clauses. They are similar in structure. The conjunction serves as the subject or the object in the sub clause. The differences lie in the conjunction words, the role in a sentence and the unification of ***FS***.

```
attri_ss --> attri_conj, np, ss_vp.  % A simple attributive clause :
                                     that I like
attri_ss --> attri_conj, vp.         % A simple attributive clause :
                                     that hates the dog

osb_ss --> osb_conj, np, ss_vp.      % A simple object/subject clause :
                                     that I like
osb_ss --> osb_conj, vp.             % A simple object/subject clause :
                                     that likes the dog
```

*__ss_vp__ is the structure of **vp** for sub-clauses. It is defined by omitting one **np** in every **vp** each time.

For attributive clauses, we use **that** as the conjunction. Since the attributive clause serves as the complement of **np**, we just add the rules:

```
attri_conj --> [that].
np(FS) --> det(FS), n(FS), attri_ss(FS).
```

We can see that the attributive clause also has a ***FS***. It will be used when **that** is the subject of the sub-clause. Otherwise, it will be omitted as we don't define the label for the object in the ***FS*** of **v**.

For subject/object clauses, we use **who** and **what** as the conjunction. They have a ***FS*** containing **is_person** to indicate whether it replaces somebody or something. Since they are actually the same except that they are different components in a sentence, we just add the rules:

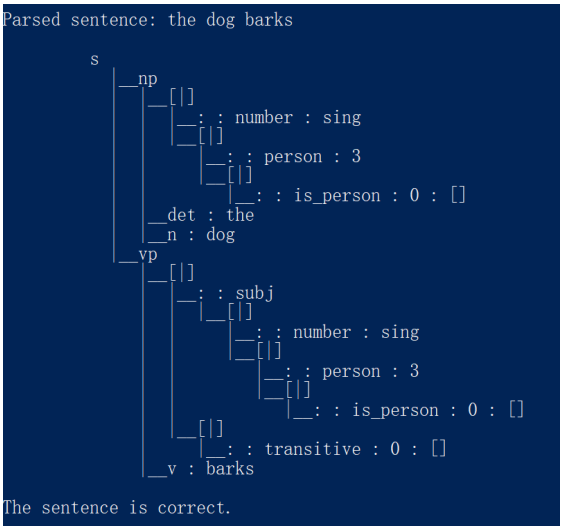
```
osb_conj([is_person:1]) --> [who].
osb_conj([is_person:0]) --> [what].
np(FS) --> osb_ss(FS).
```

As you can see, like attributive clauses, the object/subject clause also has a ***FS***. When the conjunction is an object component of a sentence, we manually define the ***FS*** of **osb_ss** as **[number:1,person:3,is_person:Isp]**, while **Isp** depends on the conjunction. In the other case where the conjunction is a subject, we copy the **FS** of the **vp([subj:*FS*|_])**. Also we need to check if the **vp** is an action that can be finished by the conjunction (the unification of **is_person** in **osb_conj** and **vp([subj:*FS*|_])**).

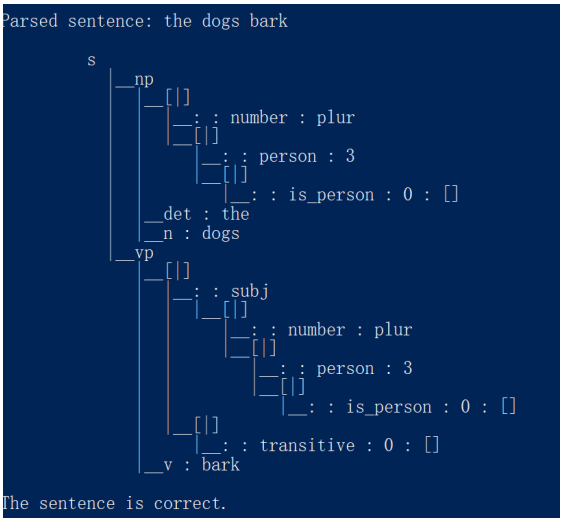
IV. Results

In this part, we will show some examples that are successfully parsed by the Chart Parser.

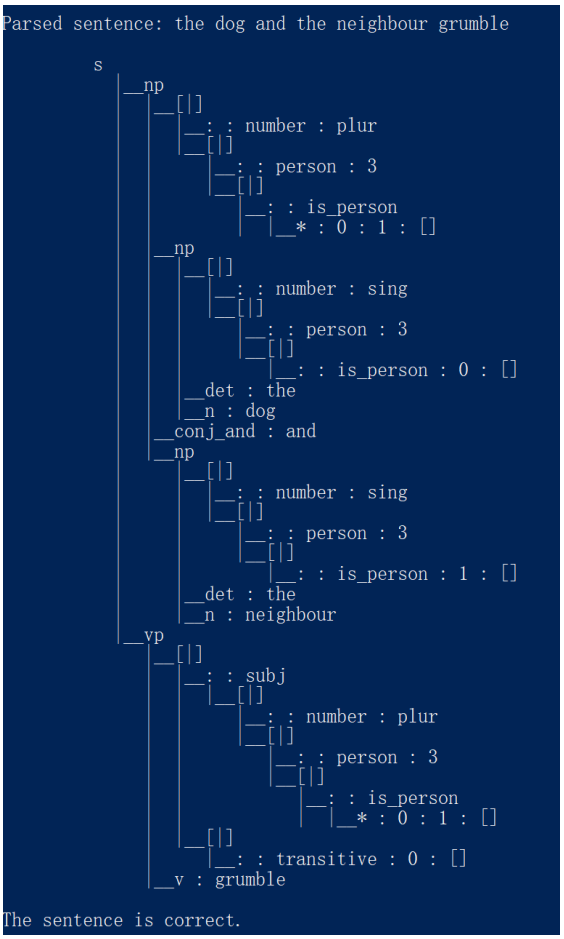
Test case 1



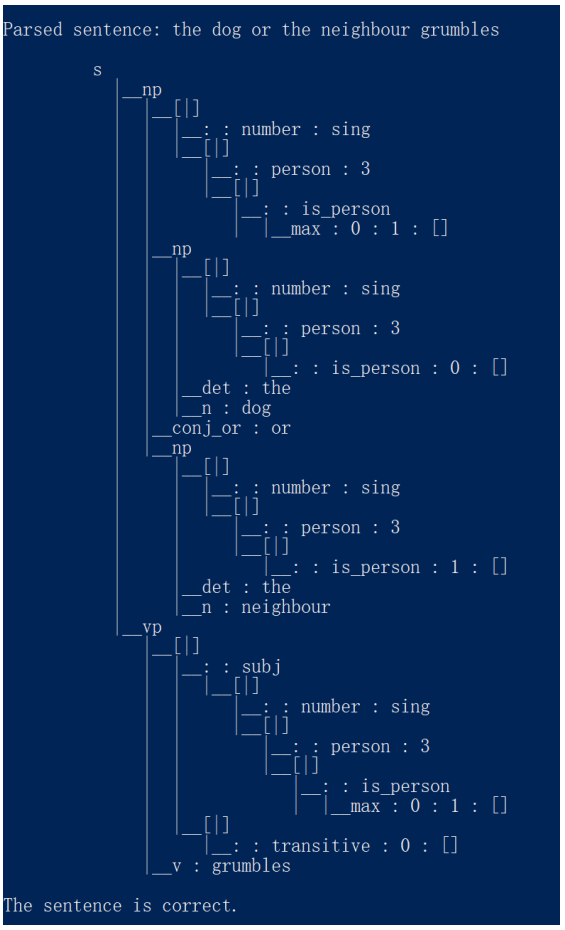
Test case 2



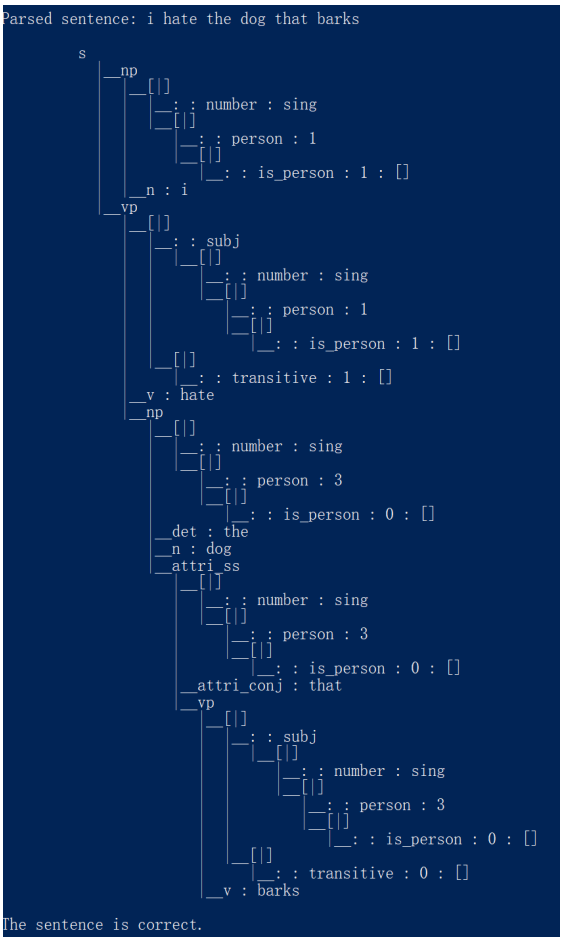
Test case 3



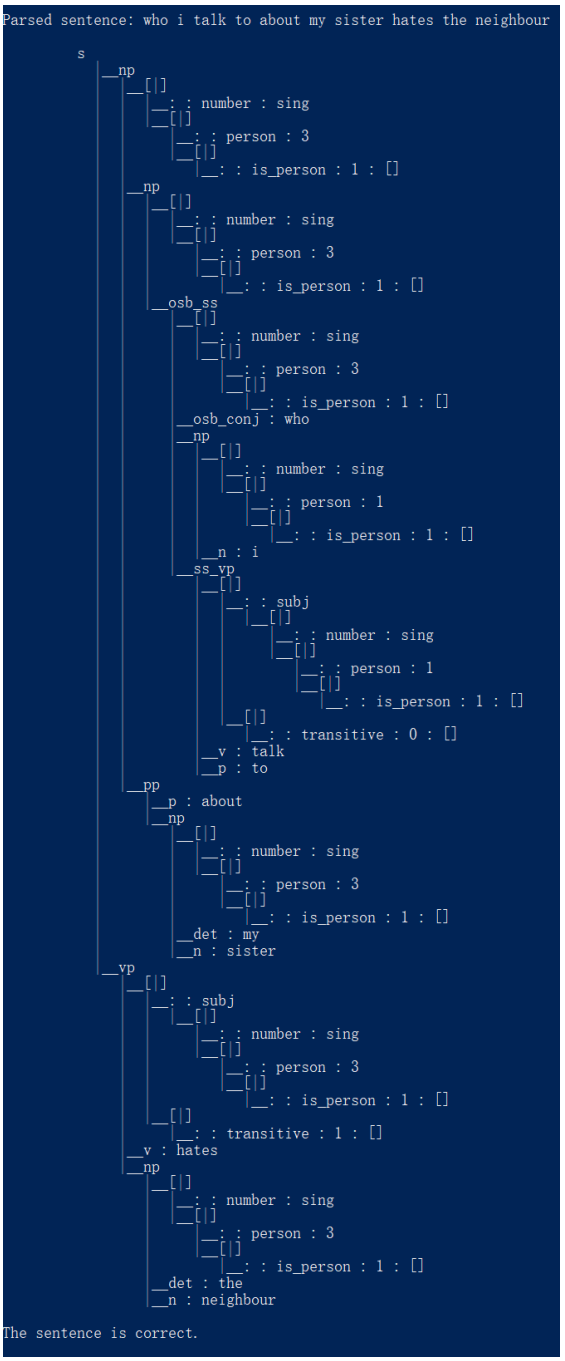
Test case 4



Test case 5



Test case 6



Test case 7

**The example sentences above are all successfully parsed and you can try other sentences on your own.*

V. Discussion

In this micro-project, we have succeeded in adding conjunctions, and sub-clauses into the grammar. Also, the Chart Parser is now able to detect some basic grammar rules and some simple meaning mistakes (like the sentence “my sister barks”). Our final result is quite close to our expectation.

However, since the program of chart parser we are using is not able to detect the predicates in curly brackets, we can only force unification by enumerating all the terms in **FS**. This method is not so elegant. Also, our **FS** is rather simple. We are just adding new properties to the grammar base.This makes the system lack extensibility and less readable.

VI. Bibliography

1. Feature Structures and Unification Grammars 11-711 Algorithms for NLP(2017). Retrieved from: http://www.cs.cmu.edu/afs/cs/user/tbergkir/www/11711fa17/Feature%20structures%20and%20unification%20F17.pdf
2. Attributive Clauses. Retrieved from: https://studfile.net/preview/4536508/page:34/