COMP 6751 Natural Language Analysis

Project 2 Report 2 (Demo)

Student: Yixuan Li 40079830

Table of Contents

I. Input and Outputs *3*

1. Test case 1: sentence example 1 3
   Test case 1 result explanation: 3
2. Test case 2: sentence example 2 4
   Test case 2 result explanation: 5
3. Test case 3: sentence example 3 5
   Test case 3 result explanation: 6
4. Test case 4: sentence example 4 7
   Test case 4 result explanation: 8
5. Test case 5: sentence example 5 9
   Test case 5 result explanation: 10
6. Test case 6: sentence example 6 11
   Test case 6 result explanation: 13
7. Test case 7: sentence example 7 14
   Test case 7 result explanation: 16
8. Test case 8: sentence example 8 17
   Test case 8 result explanation: 21
9. Test case 9: sentence example 9 23
   Test case 9 result explanation: 30
10. Test case 10: challenge text 1 32
    Challenge text 1 result explanation: 32
11. Test case 11: challenge text 2 33
    Challenge text 2 result explanation: 34
12. Test case 12: challenge text 3 35
    Challenge text 3 result explanation: 36
13. Test case 13: challenge text 4 38
    Challenge text 4 result explanation: 39
14. Test case 14: challenge text 5 41
    Challenge text 5 result explanation: 41
15. Test case 15: challenge text 6 43
    Challenge text 6 result explanation: 43
16. Test case 16: limitation example 44
    limitation text result explanation: 48

Expectations of originality:

I, student 40079830, certify that this submission is my original work and meets the Faculty's Expectations of Originality.

Date: October 27, 2020

I. Input and Outputs

1. Test case 1: sentence example 1

	Input file and content	Output on console and parse tree diagram	
Input file	data/sent1.txt	Sentences splitting results: ['John ate an apple.'] --- Part-of-speech tagging results: [[('John', 'NNP'), ('ate', 'VBD'), ('an', 'DT'), ('apple', 'NN')]] Name entities: {'John'} --- Parsing results: (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (VP[NUM=?n, TENSE='past'] (TV[TENSE='past'] ate) (NP[NUM='sg'] (DT[NUM='sg'] (Det[NUM='sg'] an)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))))	Console Output
content	John ate an apple.	There is 1 parse tree, and please see the diagram below.	Parse Tree Diagram

Test case 1 result explanation:

1) Sentence splitting

There is 1 sentence in total in sent1.txt, and the result of sentence splitting matches, showing below (copied from the table above) :

```
Sentences splitting results:
['John ate an apple.']
```

2) Named Entity Module

I use the result of “main_stan.py” in named entity module for sent1.txt.

```
Name entities:
{'John'}
```

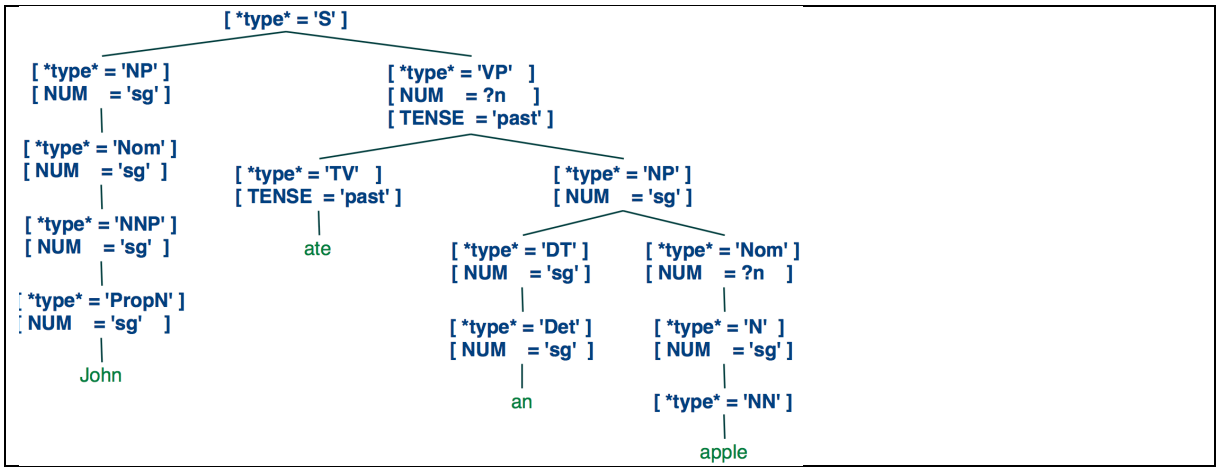
3) POS tagging

I use the result of “main_stan.py” in part-of-speech module for sent1.txt.

```
Part-of-speech tagging results:
[[('John', 'NNP'), ('ate', 'VBD'), ('an', 'DT'), ('apple', 'NN')]]
```

4) Earley Parser parsing

There is only 1 parse tree for sent1.txt showing below.



2. Test case 2: sentence example 2

In the results below, I marked the correct parse tree's result printed on the console as green and marked wrong parse tree results as bright blue.

	Input file and content	Output on console and parse tree diagram	
Input file	data/sent2.txt	Sentences splitting results: ['John ate the apple at the table.'] --- Part-of-speech tagging results: [[('John', 'NNP'), ('ate', 'VBD'), ('the', 'DT'), ('apple', 'NN'), ('at', 'IN'), ('the', 'DT'), ('table', 'NN')]] Name entities: {'John'} --- Parsing results: (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (VP[NUM=?n, TENSE='past'] (VP[NUM=?n, TENSE='past'] (TV[TENSE='past'] ate) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))) (PP[] (IN[] (P[] at)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] table))))))) (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (VP[NUM=?n, TENSE='past'] (TV[TENSE='past'] ate) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))) (PP[] (IN[] (P[] at)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] table)))))))))	Console Output

content	John ate the apple at the table.	There is 2 parse tree, and please see the diagram below.	Parse Tree Diagram

Test case 2 result explanation:

1) Sentence splitting

There is 1 sentence in total in sent2.txt, and the result matches, showing below.

```
Sentences splitting results:
['John ate the apple at the table.']
```

2) Named Entity Module

I use the result of “main_stan.py” in named entity module for sent2.txt.

```
Name entities:
{'John'}
```

3) POS tagging

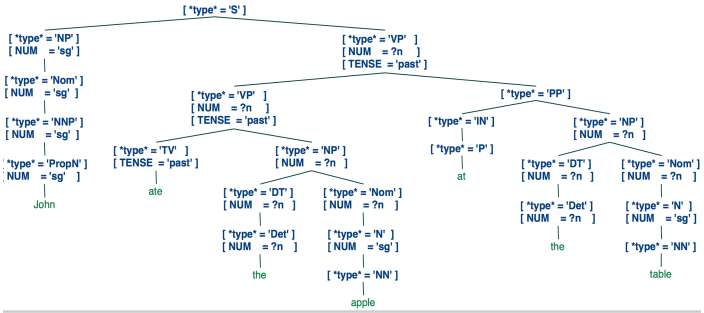
I use the result of “main_stan.py” in part-of-speech module for sent2.txt.

```
Part-of-speech tagging results:
[[('John', 'NNP'), ('ate', 'VBD'), ('the', 'DT'), ('apple', 'NN'), ('at', 'IN'), ('the', 'DT'), ('table', 'NN')]]
```

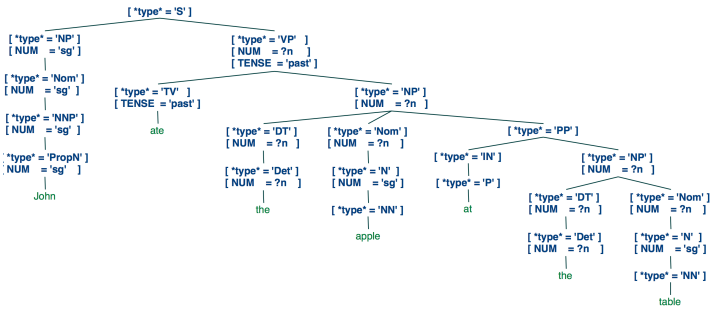
4) Earley Parser parsing

Due to the PP attachment ambiguity, there are 2 possible parse trees by my grammar.

Parse tree 1:



Parse tree 2:



But based on the meaning of the sentence, we know the PP constituent ("at the table”) more possibly attach to noun phrase “the apple” rather than the verb “ate”, **so the parse tree 2 is likely the correct one.**

3. Test case 3: sentence example 3

	Input file and content	Output on console and parse tree diagram	

Input file	data/sent3.txt	Sentences splitting results: ['On Monday, John ate the apple in the fridge.'] --- Part-of-speech tagging results: [[('On', 'IN'), ('Monday', 'NNP'), (',', ','), ('John', 'NNP'), ('ate', 'VBD'), ('the', 'DT'), ('apple', 'NN'), ('in', 'IN'), ('the', 'DT'), ('fridge', 'NN')]] Name entities: {'John'} --- Parsing results: (S[] (PP[] (IN[] (P[] On)) (DATE[] (WEEK[] Monday))) (COMMA[] ,) (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (VP[NUM=?n, TENSE='past'] (VP[NUM=?n, TENSE='past'] (TV[TENSE='past'] ate) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge))))))) (S[] (PP[] (IN[] (P[] On)) (DATE[] (WEEK[] Monday))) (COMMA[] ,) (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (VP[NUM=?n, TENSE='past'] (TV[TENSE='past'] ate) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge))))))))	Console Output
content	On Monday, John ate the apple in the fridge.	There is 2 parse tree, and please see the diagram below.	Parse Tree Diagram

Test case 3 result explanation:

1) Sentence splitting

There is 1 sentence in total in sent3.txt, and the result matches.

```
Sentences splitting results:
['On Monday, John ate the apple in the fridge.']
```

2) Named Entity Module

I use the result of “main_stan.py” in named entity module for sent3.txt.

```
Name entities:
{'John'}
```

3) POS tagging

I use the result of “main_stan.py” in part-of-speech module for sent3.txt.

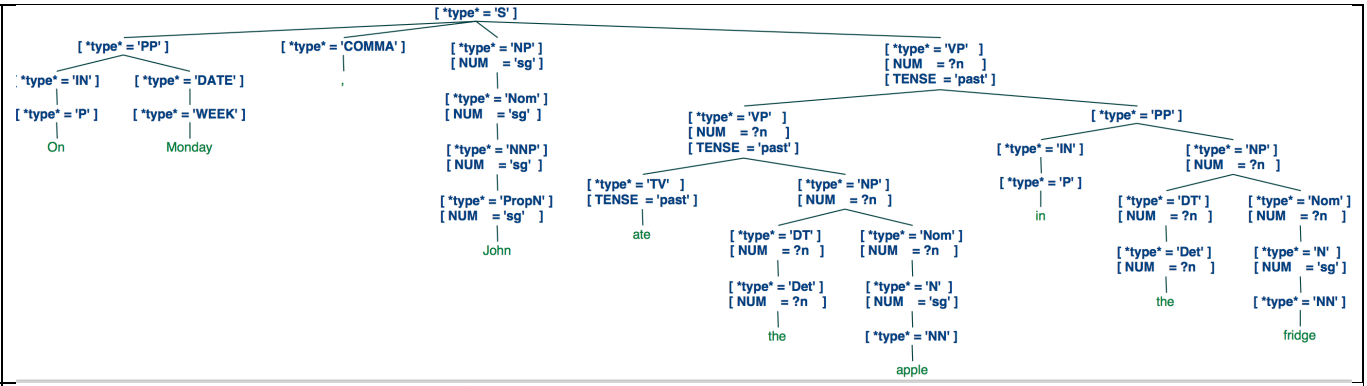
```
Part-of-speech tagging results:
[[('On', 'IN'), ('Monday', 'NNP'), (',', ','), ('John', 'NNP'), ('ate', 'VBD'), ('the', 'DT'), ('apple', 'NN'), ('in', 'IN'), ('the', 'DT'), ('fridge', 'NN')]]
```

4) Earley Parser parsing

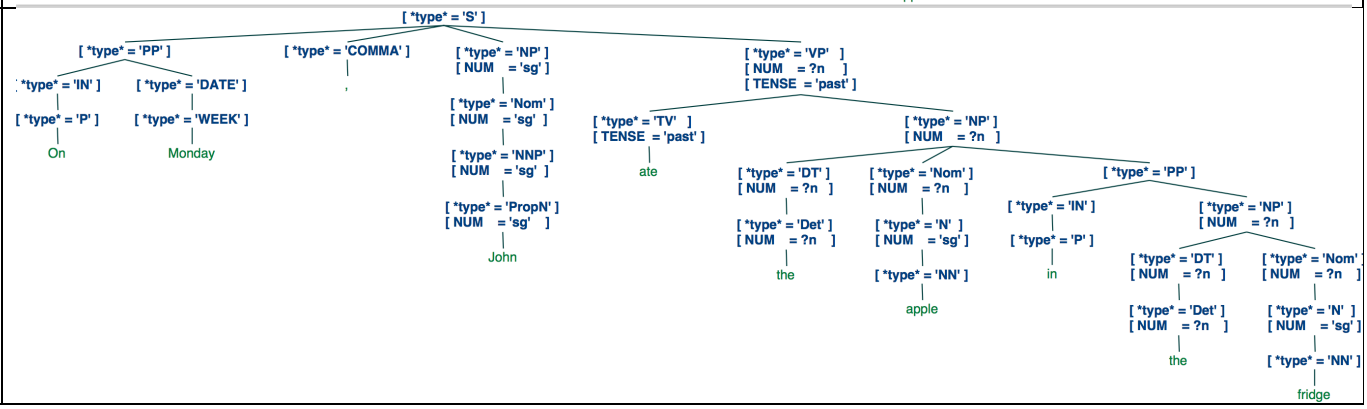
Due to the PP attachment ambiguity, my grammar generates 2 possible parse trees.

In Parse tree 1, the PP constituent “in the fridge” attaches to the verb “ate”, and in the parse tree 2, the PP attaches to the noun “apple”.

Parse tree 1:



Parse tree 2:



Based on the meaning of the sentence**, the parse tree 2 is the more proper one**.

4. Test case 4: sentence example 4

	Input file and content	Output on console and parse tree diagram	

Input file	data/sent4.txt	Sentences splitting results: ['On Monday, John ate the apple in his office.'] --- Part-of-speech tagging results: [[('On', 'IN'), ('Monday', 'NNP'), (',', ','), ('John', 'NNP'), ('ate', 'VBD'), ('the', 'DT'), ('apple', 'NN'), ('in', 'IN'), ('his', 'PRP$'), ('office', 'NN')]] Name entities: {'John'} --- Parsing results: (S[] (PP[] (IN[] (P[] On)) (DATE[] (WEEK[] Monday))) (COMMA[] ,) (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (VP[NUM=?n, TENSE='past'] (VP[NUM=?n, TENSE='past'] (TV[TENSE='past'] ate) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (PRP[] his) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] office)))))))) (S[] (PP[] (IN[] (P[] On)) (DATE[] (WEEK[] Monday))) (COMMA[] ,) (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (VP[NUM=?n, TENSE='past'] (TV[TENSE='past'] ate) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (PRP[] his) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] office))))))))	Console Output
content	On Monday, John ate the apple in his office.	There are 2 parse trees generated, please see the diagrams below.	Parse Tree Diagram

Test case 4 result explanation:

1) Sentence splitting

There is 1 sentence in total in sent4.txt, and the result matches.

```
Sentences splitting results:
['On Monday, John ate the apple in his office.']
```

2) Named Entity Module

I use the result of “main_stan.py” in named entity module for sent4.txt.

```
Name entities:
{'John'}
```

3) POS tagging

I use the result of “main_stan.py” in part-of-speech module for sent4.txt.

```
Part-of-speech tagging results:
[[('On', 'IN'), ('Monday', 'NNP'), (',', ','), ('John', 'NNP'), ('ate', 'VBD'), ('the', 'DT'), ('apple', 'NN'), ('in', 'IN'), ('his', 'PRP$'), ('office', 'NN')]]
```

4) Earley Parser parsing

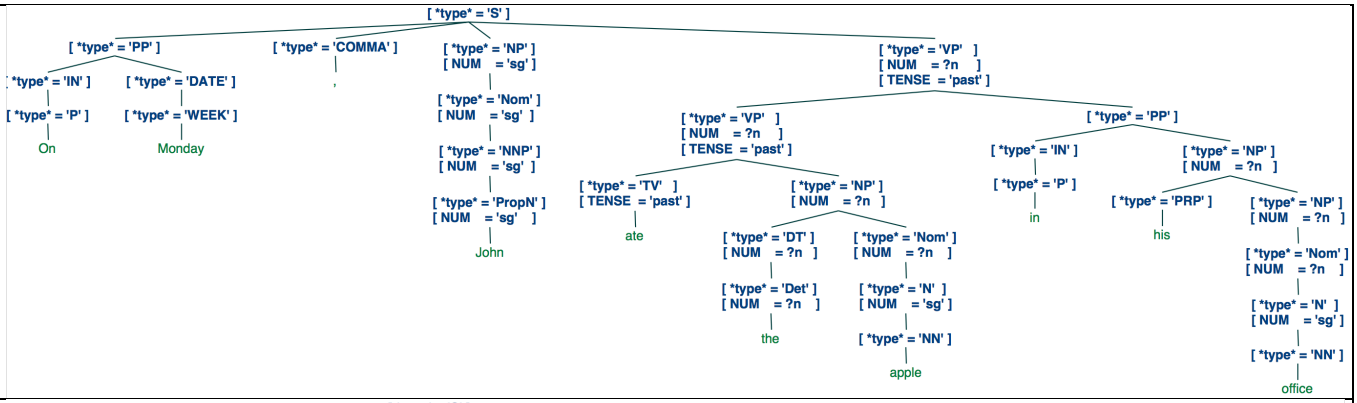
Due to the PP attachment ambiguity, my grammar generates 2 possible parse trees.

In the parse tree 1, the PP constituent attaches to the Verb Phrase, meaning *John sitting in his office ate the apple*;

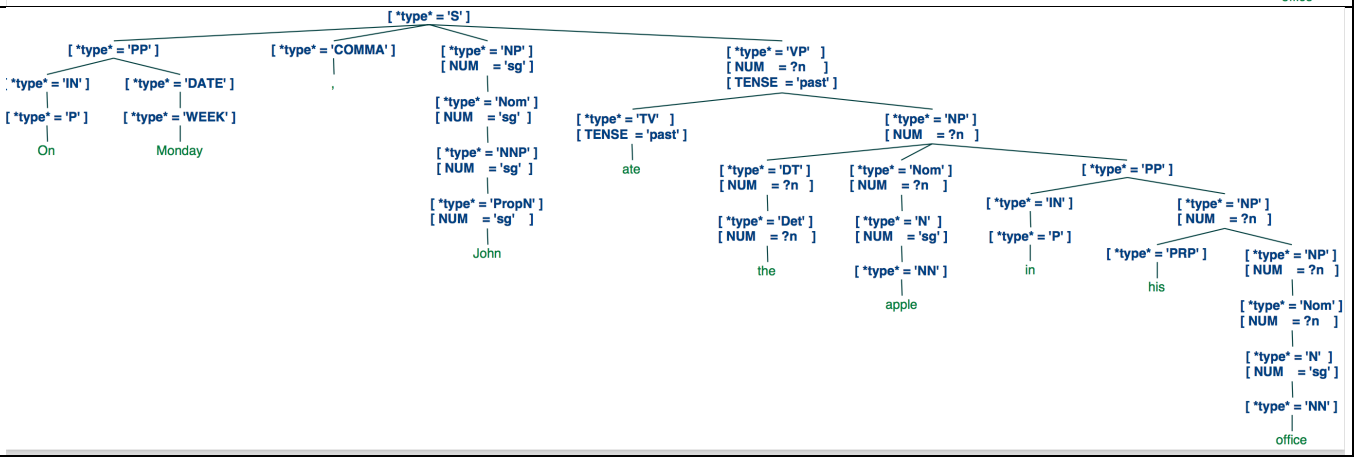
In the parse tree 2, it modifies the NOM “apple”, *meaning the apple which John ate was in his office*.

Based on the meaning of the sentence, **both parse trees should be acceptable**.

Parse tree 1:



Parse tree 2:



5. Test case 5: sentence example 5

	Input file and content	Output on console and parse tree diagram	

Input file	data/sent5.txt	Sentences splitting results: ['On Monday, John ate refrigerator apple in his office.'] --- Part-of-speech tagging results: [[('On', 'IN'), ('Monday', 'NNP'), (',', ','), ('John', 'NNP'), ('ate', 'VBD'), ('refrigerator', 'NN'), ('apple', 'NN'), ('in', 'IN'), ('his', 'PRP$'), ('office', 'NN')]] Name entities: {'John'} --- Parsing results: (S[] (PP[] (IN[] (P[] On)) (DATE[] (WEEK[] Monday))) (COMMA[] ,) (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (VP[NUM=?n, TENSE='past'] (VP[NUM=?n, TENSE='past'] (TV[TENSE='past'] ate) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] refrigerator)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple)))))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (PRP[] his) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] office)))))))) (S[] (PP[] (IN[] (P[] On)) (DATE[] (WEEK[] Monday))) (COMMA[] ,) (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (VP[NUM=?n, TENSE='past'] (TV[TENSE='past'] ate) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] refrigerator)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple)))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (PRP[] his) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] office)))))))))	Console Output
content	On Monday, John ate refrigerator apple in his office.	There are 2 parse trees generated, please see the diagrams below.	Parse Tree Diagram

Test case 5 result explanation:

1) Sentence splitting

There is 1 sentence in total in sent5.txt, and the result matches.

```
Sentences splitting results:
['On Monday, John ate refrigerator apple in his office.']
```

2) Named Entity Module

I use the result of “main_stan.py” in named entity module for sent5.txt

Name entities:
{'John'}

3) POS tagging

I use the result of “main_stan.py” in part-of-speech module for sent5.txt

Part-of-speech tagging results:
[[('On', 'IN'), ('Monday', 'NNP'), (',', ','), ('John', 'NNP'), ('ate', 'VBD'), ('refrigerator', 'NN'), ('apple', 'NN'), ('in', 'IN'), ('his', 'PRP$'), ('office', 'NN')]]

4) Earley Parser parsing

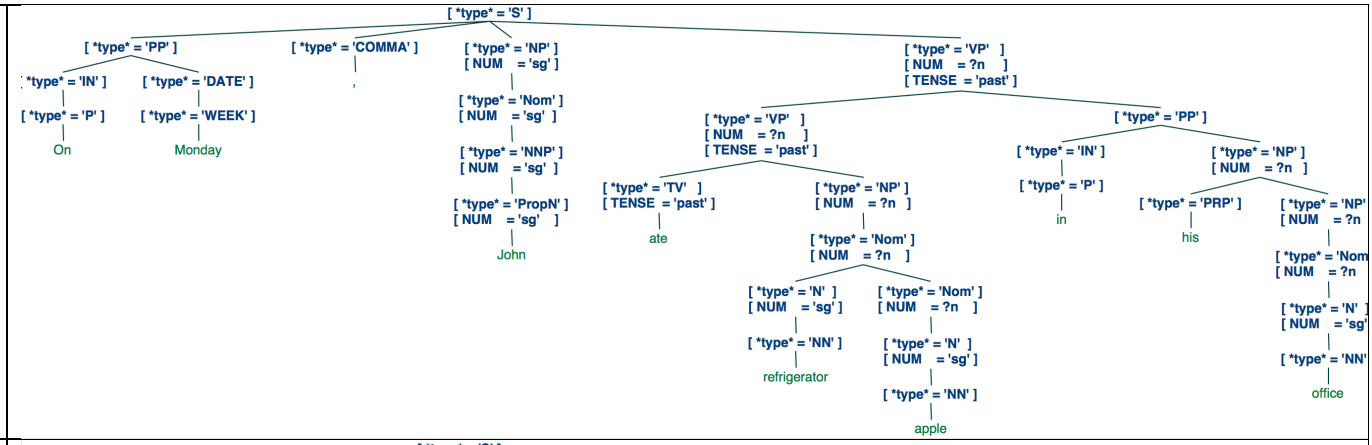
Due to the PP attachment ambiguity, my grammar generates 2 possible parse trees.

In the parse tree 1, the PP constituent “in his office” attaches to the Verb Phrase” meaning John sitting in his office ate the refrigerator apple;

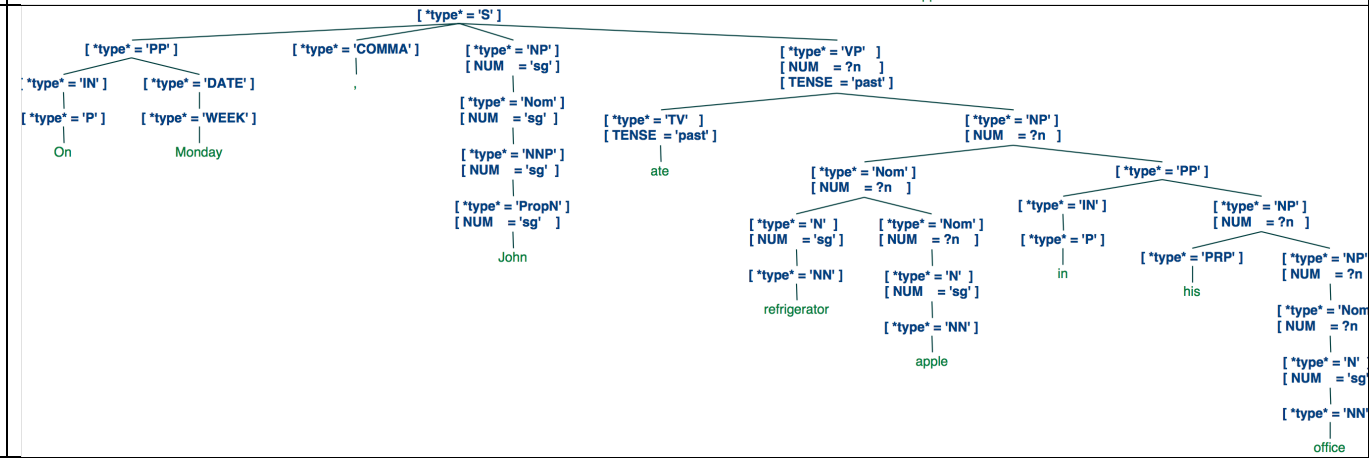
In the parse tree 2, the PP constituent “in his office” modifies the NOM “refrigerator apple” meaning the refrigerator apple that John ate was in his office.

Based on the meaning of the sentence, **both parse trees could be possible**.

Parse tree 1:



Parse tree 2:



6. Test case 6: sentence example 6

	Input file and content	Output on console and parse tree diagram	

Input file

data/sent6.txt

Console Output

```
Sentences splitting results:
['Last week, on Monday, John finally took the apple from the fridge to his office.']
--------------------------------------------------
Part-of-speech tagging results:
[[('Last', 'JJ'), ('week', 'NN'), (',', ','), ('on', 'IN'), ('Monday', 'NNP'), (',', ','), ('John', 'NNP'), ('finally', 'RB'), ('took', 'VBD'),
('the', 'DT'), ('apple', 'NN'), ('from', 'IN'), ('the', 'DT'), ('fridge', 'NN'), ('to', 'IN'), ('his', 'PRP$'), ('office', 'NN')]]
Name entities:
{'John'}
--------------------------------------------------
Parsing results:
(S[]
  (DATE[] (JJ[] Last) (WEEK[] week))
  (COMMA[] ,)
  (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Monday)))
  (COMMA[] ,)
  (NP[NUM='sg']
    (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John))))
  (VP[NUM=?n, TENSE='past']
    (VP[NUM=?n, TENSE='past']
      (VP[NUM=?n, TENSE='past']
        (RB[] finally)
        (TV[TENSE='past'] took)
        (NP[NUM=?n]
          (DT[NUM=?n] (Det[NUM=?n] the))
          (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple)))))
      (PP[]
        (IN[] (P[] from))
        (NP[NUM=?n]
          (DT[NUM=?n] (Det[NUM=?n] the))
          (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge))))))
    (PP[]
      (IN[] (P[] to))
      (NP[NUM=?n]
        (PRP[] his)
        (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] office)))))))))
(S[]
  (DATE[] (JJ[] Last) (WEEK[] week))
  (COMMA[] ,)
  (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Monday)))
  (COMMA[] ,)
  (NP[NUM='sg']
    (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John))))
  (VP[NUM=?n, TENSE='past']
    (VP[NUM=?n, TENSE='past']
      (VP[NUM=?n, TENSE='past']
        (RB[] finally)
        (TV[TENSE='past'] took)
        (NP[NUM=?n]
          (DT[NUM=?n] (Det[NUM=?n] the))
          (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple)))))
      (PP[]
        (IN[] (P[] from))
        (NP[NUM=?n]
          (DT[NUM=?n] (Det[NUM=?n] the))
          (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge)))))))
    (PP[]
      (IN[] (P[] to))
      (NP[NUM=?n]
        (PRP[] his)
        (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] office)))))))))
(S[]
  (DATE[] (JJ[] Last) (WEEK[] week))
  (COMMA[] ,)
  (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Monday)))
  (COMMA[] ,)
  (NP[NUM='sg']
    (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John))))
  (VP[NUM=?n, TENSE='past']
    (VP[NUM=?n, TENSE='past']
      (VP[NUM=?n, TENSE='past']
        (RB[] finally)
        (TV[TENSE='past'] took)
        (NP[NUM=?n]
          (DT[NUM=?n] (Det[NUM=?n] the))
          (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple)))))
      (PP[]
        (IN[] (P[] from))
        (NP[NUM=?n]
          (DT[NUM=?n] (Det[NUM=?n] the))
          (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge)))))))
    (PP[]
      (IN[] (P[] to))
      (NP[NUM=?n]
        (PRP[] his)
        (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] office))))))))))
(S[]
  (DATE[] (JJ[] Last) (WEEK[] week))
  (COMMA[] ,)
  (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Monday)))
  (COMMA[] ,)
  (NP[NUM='sg']
    (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John))))
  (VP[NUM=?n, TENSE='past']
    (RB[] finally)
    (TV[TENSE='past'] took)
    (NP[NUM=?n]
      (DT[NUM=?n] (Det[NUM=?n] the))
      (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple)))
      (PP[]
        (IN[] (P[] from))
        (NP[NUM=?n]
          (DT[NUM=?n] (Det[NUM=?n] the))
```

		(Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge))) (PP[] (IN[] (P[] to)) (NP[NUM=?n] (PRP[] his) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] office)))))))))))	
content	Last week, on Monday, John finally took the apple from the fridge to his office.	There are 4 parse trees generated, please see the diagrams below.	Parse Tree Diagram

Test case 6 result explanation:

1) Sentence splitting

There is 1 sentence in total in sent6.txt, and the result matches.

```
Sentences splitting results:
['Last week, on Monday, John finally took the apple from the fridge to his office.']
```

2) Named Entity Module

I use the result of “main_stan.py” in named entity module for sent6.txt

```
Name entities:
{'John'}
```

3) POS tagging

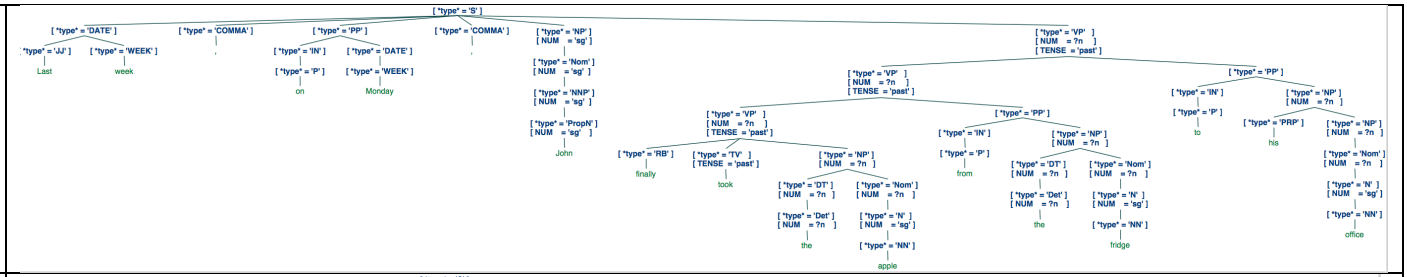
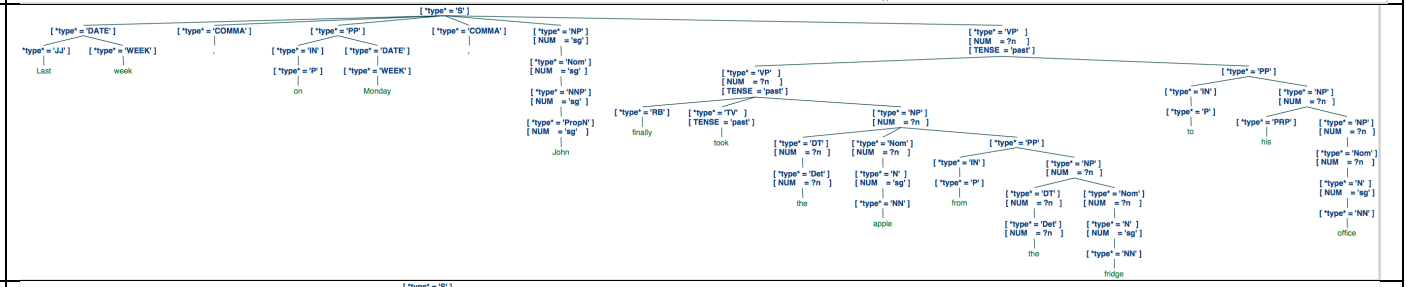
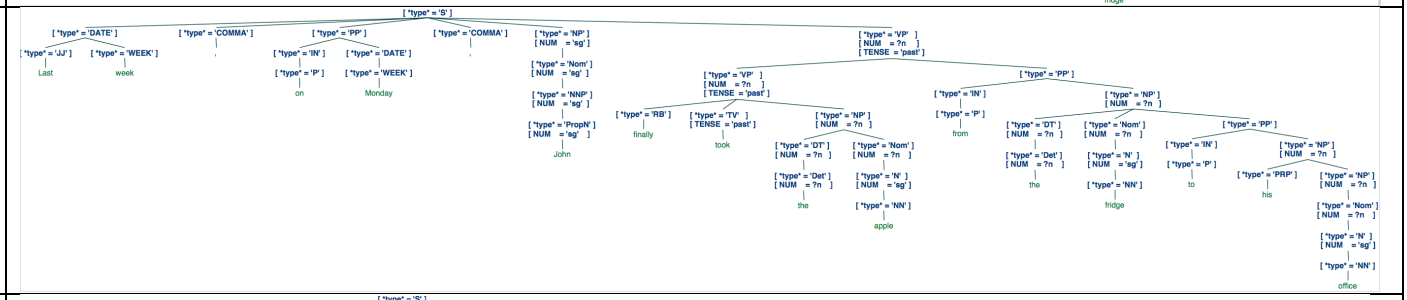
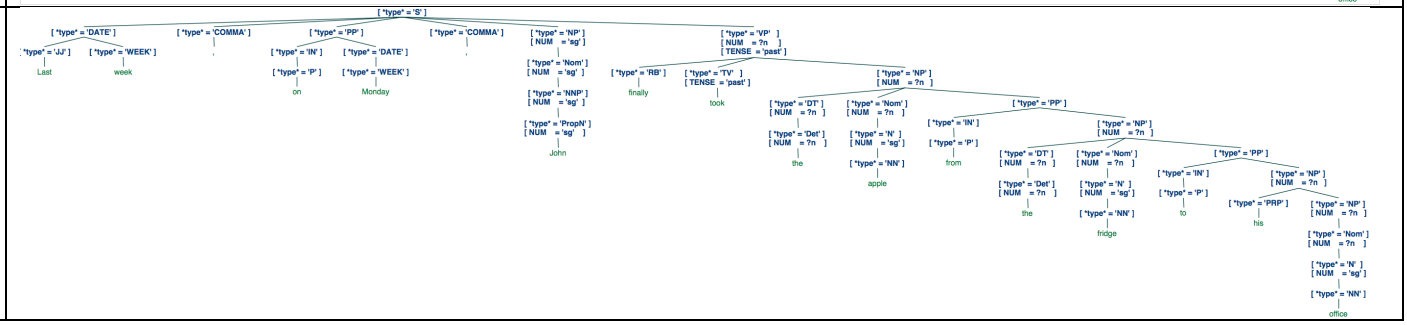
I use the result of “main_stan.py” in part-of-speech module for sent6.txt

```
Part-of-speech tagging results:
[[('Last', 'JJ'), ('week', 'NN'), (',', ','), ('on', 'IN'), ('Monday', 'NNP'), (',', ','), ('John', 'NNP'), ('finally', 'RB'), ('took', 'VBD'),
('the', 'DT'), ('apple', 'NN'), ('from', 'IN'), ('the', 'DT'), ('fridge', 'NN'), ('to', 'IN'), ('his', 'PRP$'), ('office', 'NN')]]
```

4) Earley Parser parsing

Because of the fact that “from the fridge” and “to his office” are both PP constituents, it creates a nested PP attachment ambiguity problem. The nested PP ambiguity means that the second PP constituent might be the modifier of the NOM in the first PP constituent, i.e. “the fridge” (result as Parse tree 3), so my grammar generates 4 parse trees in total.

Based on the meaning, “from the fridge” and “to his office” should both attach to the Verb “took”, **so Parse tree 1 is the correct one among the results.**

Parse tree 1	
Parse tree 2	
Parse tree 3	
Parse tree 4	

7. Test case 7: sentence example 7

	Input file and content	Output on console and parse tree diagram	

Input file			Console Output
	data/sent7.txt	Sentences splitting results: ['Last Monday, John promised that he will put an apple in the fridge.', 'He will eat it on Tuesday at his desk.', 'It will be crunchy.'] --- Part-of-speech tagging results: [[('Last', 'JJ'), ('Monday', 'NNP'), (',', ','), ('John', 'NNP'), ('promised', 'VBD'), ('that', 'IN'), ('he', 'PRP'), ('will', 'MD'), ('put', 'VB'), ('an', 'DT'), ('apple', 'NN'), ('in', 'IN'), ('the', 'DT'), ('fridge', 'NN')], [('He', 'PRP'), ('will', 'MD'), ('eat', 'VB'), ('it', 'PRP'), ('on', 'IN'), ('Tuesday', 'NNP'), ('at', 'IN'), ('his', 'PRP$'), ('desk', 'NN')], [('It', 'PRP'), ('will', 'MD'), ('be', 'VB'), ('crunchy', 'JJ')]] Name entities: {'John'} --- Parsing results: (S[] (DATE[] (JJ[] Last) (WEEK[] Monday)) (COMMA[] ,) (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (TV[TENSE='past'] promised) (INTRO[] that) (S[] (NP[NUM=?n] (PRP[] he)) (VP[NUM=?n, TENSE='inf'] (VP[NUM=?n, TENSE='inf'] (MD[TENSE='inf'] will) (TV[TENSE='inf'] put) (NP[NUM='sg'] (DT[NUM='sg'] (Det[NUM='sg'] an)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge)))))))) (S[] (DATE[] (JJ[] Last) (WEEK[] Monday)) (COMMA[] ,) (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (TV[TENSE='past'] promised) (INTRO[] that) (S[] (NP[NUM=?n] (PRP[] he)) (VP[NUM=?n, TENSE='inf'] (MD[TENSE='inf'] will) (TV[TENSE='inf'] put) (NP[NUM='sg'] (DT[NUM='sg'] (Det[NUM='sg'] an)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge))))))))) (S[] (NP[NUM=?n] (PRP[] He)) (VP[NUM=?n, TENSE='inf'] (VP[NUM=?n, TENSE='inf'] (VP[NUM=?n, TENSE='inf'] (MD[TENSE='inf'] will) (TV[TENSE='inf'] eat) (NP[NUM=?n] (PRP[] it))) (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Tuesday)))) (PP[] (IN[] (P[] at)) (NP[NUM=?n] (PRP[] his) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] desk)))))))) (S[] (NP[NUM=?n] (PRP[] It)) (VP[NUM=?n, TENSE='inf'] (MD[TENSE='inf'] will) (AUX[TENSE='inf'] be) (JJ[] crunchy)))	

content	Last Monday, John promised that he will put an apple in the fridge. He will eat it on Tuesday at his desk. It will be crunchy.	There are 4 parse trees generated, please see the diagrams below.	Parse Tree Diagram

Test case 7 result explanation:

1) Sentence splitting

There is 3 sentences in total in sent7.txt, and the result matches.

Sentences splitting results:
['Last Monday, John promised that he will put an apple in the fridge.', 'He will eat it on Tuesday at his desk.', 'It will be crunchy.']

2) Named Entity Module

I use the result of “main_stan.py” in named entity module for sent7.txt.

Name entities:
{'John'}

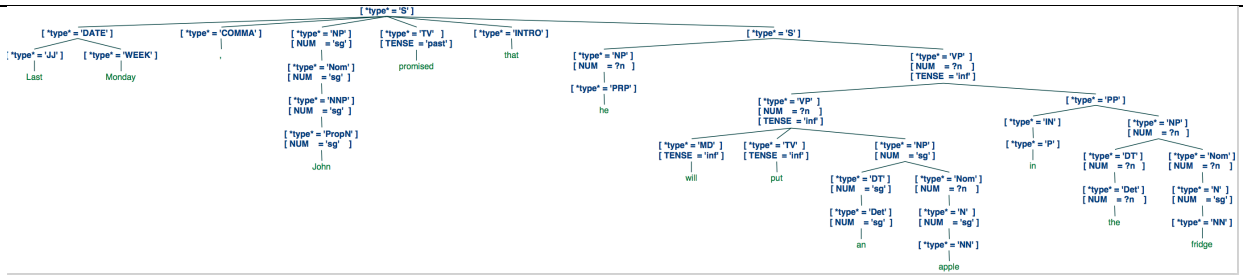
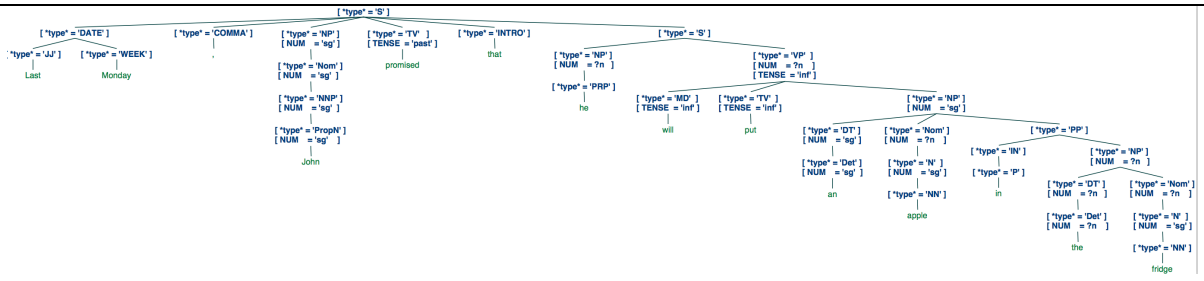
3) POS tagging

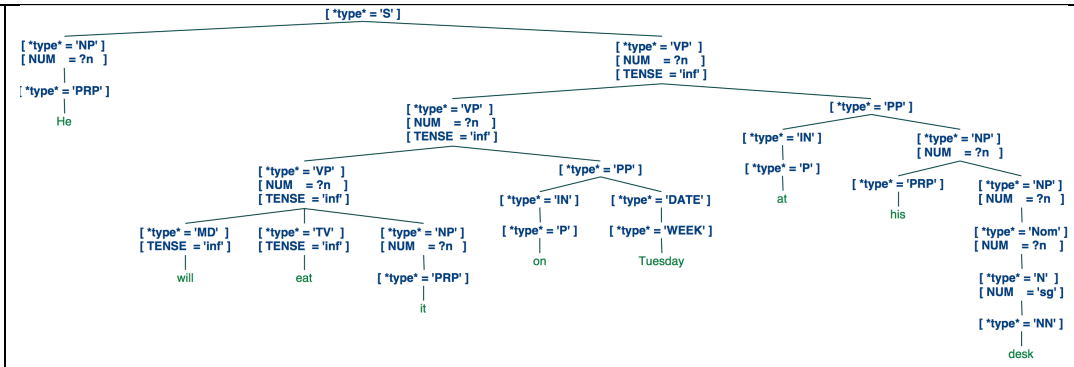
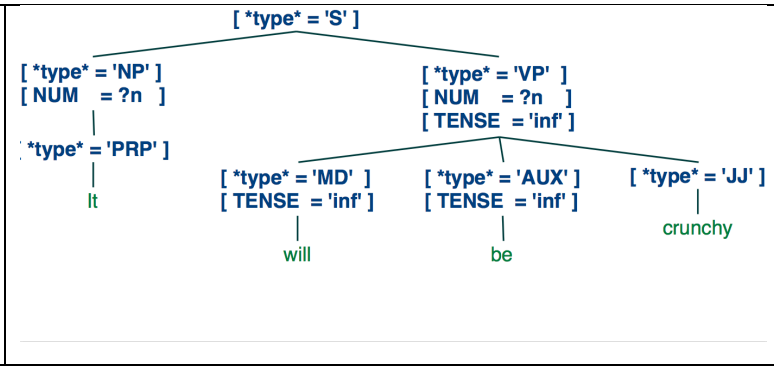
I use the result of “main_stan.py” in part-of-speech module for sent7.txt.

Part-of-speech tagging results:
[[('Last', 'JJ'), ('Monday', 'NNP'), (',', ','), ('John', 'NNP'), ('promised', 'VBD'), ('that', 'IN'), ('he', 'PRP'), ('will', 'MD'), ('put', 'VB'), ('an', 'DT'), ('apple', 'NN'), ('in', 'IN'), ('the', 'DT'), ('fridge', 'NN')], [('He', 'PRP'), ('will', 'MD'), ('eat', 'VB'), ('it', 'PRP'), ('on', 'IN'), ('Tuesday', 'NNP'), ('at', 'IN'), ('his', 'PRP$'), ('desk', 'NN')], [('It', 'PRP'), ('will', 'MD'), ('be', 'VB'), ('crunchy', 'JJ')]]

4) Earley Parser parsing

There are 2 possible parse trees for the first sentence. In the parse tree 1, PP constituent “in the fridge” attaches to the verb “put”; in the parse tree 2, it modifies the NOM “apple”, and based on the meaning, **parse tree 1 is the correct parse tree for sentence 1.**

Sentence 1: Last Monday, John promised that he will put an apple in the fridge.	
Parse tree 1	
Parse tree 2	
Sentence 2: He will eat it on Tuesday at his desk.	

Parse tree	
Sentence 3: It will be crunchy.	
Parse tree	

8. Test case 8: sentence example 8

	Input file and content	Output on console and parse tree diagram	

Input file			Console Output
	data/sent8.txt	Sentences splitting results: ["On Monday, September 17, 2018, John O'Malley promised his colleague Mary that he would put a replacement apple in the office fridge.", "O'Malley intended to share it with her on Tuesday at his desk and anticipated that the crunchy treat would delight them both.", 'But she was sick that day.'] --- Part-of-speech tagging results: [[('On', 'IN'), ('Monday', 'NNP'), (',', ','), ('September', 'NNP'), ('17', 'CD'), (',', ','), ('2018', 'CD'), (',', ','), ("John O'Malley", 'NNP'), ('promised', 'VBD'), ('his', 'PRP$'), ('colleague', 'NN'), ('Mary', 'NNP'), ('that', 'IN'), ('he', 'PRP'), ('would', 'MD'), ('put', 'VB'), ('a', 'DT'), ('replacement', 'NN'), ('apple', 'NN'), ('in', 'IN'), ('the', 'DT'), ('office', 'NN'), ('fridge', 'NN')], [("O'Malley", 'NNP'), ('intended', 'VBD'), ('to', 'TO'), ('share', 'VB'), ('it', 'PRP'), ('with', 'IN'), ('her', 'PRP'), ('on', 'IN'), ('Tuesday', 'NNP'), ('at', 'IN'), ('his', 'PRP$'), ('desk', 'NN'), ('and', 'CC'), ('anticipated', 'VBD'), ('that', 'IN'), ('the', 'DT'), ('crunchy', 'JJ'), ('treat', 'NN'), ('would', 'MD'), ('delight', 'VB'), ('them', 'PRP'), ('both', 'DT')], [('But', 'CC'), ('she', 'PRP'), ('was', 'VBD'), ('sick', 'JJ'), ('that', 'DT'), ('day', 'NN')]] Name entities: {'Mary', "John O'Malley", "O'Malley"} --- Parsing results: (S[] (PP[] (IN[] (P[] On)) (DATE[] (WEEK[] Monday))) (COMMA[] ,) (DATE[] (MONTH_STR[] September) (DAY[] 17) (SEP[] ,) (YEAR[] 2018)) (COMMA[] ,) (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John O'Malley)))) (TV[TENSE='past'] promised) (NP[NUM='sg'] (PRP[] his) (NP[NUM='sg'] (Nom[NUM='sg'] (N[NUM='sg'] (NN[] colleague)) (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary)))))) (INTRO[] that) (S[] (NP[NUM=?n] (PRP[] he)) (VP[NUM=?n, TENSE='past'] (VP[NUM=?n, TENSE='past'] (MD[TENSE='past'] would) (TV[TENSE='inf'] put) (NP[NUM='sg'] (DT[NUM='sg'] (Det[NUM='sg'] a)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] replacement)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple)))))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] office)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge))))))))) (S[] (PP[] (IN[] (P[] On)) (DATE[] (WEEK[] Monday))) (COMMA[] ,) (DATE[] (MONTH_STR[] September) (DAY[] 17) (SEP[] ,) (YEAR[] 2018)) (COMMA[] ,) (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John O'Malley)))) (TV[TENSE='past'] promised) (NP[NUM='sg'] (PRP[] his) (NP[NUM='sg'] (Nom[NUM='sg'] (N[NUM='sg'] (NN[] colleague))	

```
      (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary))))))
 (INTRO[] that)
 (S[]
  (NP[NUM=?n] (PRP[] he))
  (VP[NUM=?n, TENSE='past']
   (MD[TENSE='past'] would)
   (TV[TENSE='inf'] put)
   (NP[NUM='sg']
    (DT[NUM='sg'] (Det[NUM='sg'] a))
    (Nom[NUM=?n]
     (N[NUM='sg'] (NN[] replacement))
     (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))
    (PP[]
     (IN[] (P[] in))
     (NP[NUM=?n]
      (DT[NUM=?n] (Det[NUM=?n] the))
      (Nom[NUM=?n]
       (N[NUM='sg'] (NN[] office))
       (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge)))))))))
(S[]
 (NP[NUM='sg']
  (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] O'Malley))))
 (VP[NUM=?n, TENSE='past']
  (VP[NUM=?n, TENSE='past']
   (IV[TENSE='past'] intended)
   (TO[] to)
   (VP[NUM=?n, TENSE='inf']
    (VP[NUM=?n, TENSE='inf']
     (VP[NUM=?n, TENSE='inf']
      (TV[TENSE='inf'] share)
      (NP[NUM=?n] (PRP[] it)))
     (PP[] (IN[] (P[] with)) (NP[NUM=?n] (PRP[] her))))
    (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Tuesday)))))
  (PP[]
   (IN[] (P[] at))
   (NP[NUM=?n]
    (PRP[] his)
    (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] desk)))))))
 (CC[] and)
 (TV[TENSE='past'] anticipated)
 (INTRO[] that)
 (S[]
  (NP[NUM=?n]
   (DT[NUM=?n] (Det[NUM=?n] the))
   (JJ[] crunchy)
   (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] treat)))))
  (VP[NUM=?n, TENSE='past']
   (MD[TENSE='past'] would)
   (TV[TENSE='inf'] delight)
   (NP[NUM=?n] (PRP[] them))
   (RB[] both))))
(S[]
 (NP[NUM='sg']
  (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] O'Malley))))
 (VP[NUM=?n, TENSE='past']
  (VP[NUM=?n, TENSE='past']
   (VP[NUM=?n, TENSE='past']
    (VP[NUM=?n, TENSE='past']
     (IV[TENSE='past'] intended)
     (TO[] to)
     (VP[NUM=?n, TENSE='inf']
      (TV[TENSE='inf'] share)
      (NP[NUM=?n] (PRP[] it))))
    (PP[] (IN[] (P[] with)) (NP[NUM=?n] (PRP[] her))))
   (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Tuesday))))
  (PP[]
   (IN[] (P[] at))
   (NP[NUM=?n]
    (PRP[] his)
    (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] desk)))))))
 (CC[] and)
 (TV[TENSE='past'] anticipated)
 (INTRO[] that)
 (S[]
  (NP[NUM=?n]
   (DT[NUM=?n] (Det[NUM=?n] the))
   (JJ[] crunchy)
```

```
    (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] treat)))))
   (VP[NUM=?n, TENSE='past']
    (MD[TENSE='past'] would)
    (TV[TENSE='inf'] delight)
    (NP[NUM=?n] (PRP[] them))
    (RB[] both))))
(S[]
 (NP[NUM='sg']
  (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] O'Malley))))
 (VP[NUM=?n, TENSE='past']
  (VP[NUM=?n, TENSE='past']
   (VP[NUM=?n, TENSE='past']
    (IV[TENSE='past'] intended)
    (TO[] to)
    (VP[NUM=?n, TENSE='inf']
     (VP[NUM=?n, TENSE='inf']
      (TV[TENSE='inf'] share)
      (NP[NUM=?n] (PRP[] it)))
     (PP[] (IN[] (P[] with)) (NP[NUM=?n] (PRP[] her)))))
   (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Tuesday))))
  (PP[]
   (IN[] (P[] at))
   (NP[NUM=?n]
    (PRP[] his)
    (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] desk)))))))
 (CC[] and)
 (TV[TENSE='past'] anticipated)
 (INTRO[] that)
 (S[]
  (NP[NUM=?n]
   (DT[NUM=?n] (Det[NUM=?n] the))
   (JJ[] crunchy)
   (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] treat)))))
  (VP[NUM=?n, TENSE='past']
   (MD[TENSE='past'] would)
   (TV[TENSE='inf'] delight)
   (NP[NUM=?n] (PRP[] them))
   (RB[] both))))
(S[]
 (NP[NUM='sg']
  (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] O'Malley))))
 (VP[NUM=?n, TENSE='past']
  (IV[TENSE='past'] intended)
  (TO[] to)
  (VP[NUM=?n, TENSE='inf']
   (VP[NUM=?n, TENSE='inf']
    (VP[NUM=?n, TENSE='inf']
     (VP[NUM=?n, TENSE='inf']
      (TV[TENSE='inf'] share)
      (NP[NUM=?n] (PRP[] it)))
     (PP[] (IN[] (P[] with)) (NP[NUM=?n] (PRP[] her))))
    (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Tuesday))))
   (PP[]
    (IN[] (P[] at))
    (NP[NUM=?n]
     (PRP[] his)
     (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] desk))))))))
 (CC[] and)
 (TV[TENSE='past'] anticipated)
 (INTRO[] that)
 (S[]
  (NP[NUM=?n]
   (DT[NUM=?n] (Det[NUM=?n] the))
   (JJ[] crunchy)
   (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] treat)))))
  (VP[NUM=?n, TENSE='past']
   (MD[TENSE='past'] would)
   (TV[TENSE='inf'] delight)
   (NP[NUM=?n] (PRP[] them))
```

		(RB[] both)))) (S[] (CC[] But) (NP[NUM=?n] (PRP[] she)) (VP[NUM=?n, TENSE='past'] (VP[NUM=?n, TENSE='past'] (AUX[TENSE='past'] was) (JJ[] sick)) (PP[] (IN[] (P[] that)) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] day)))))))	
content	On Monday, September 17, 2018, John O'Malley promised his colleague Mary that he would put a replacement apple in the office fridge. O'Malley intended to share it with her on Tuesday at his desk and anticipated that the crunchy treat would delight them both. But she was sick that day.	There are 7 parse trees generated, please see the diagrams below.	Parse Tree Diagram

Test case 8 result explanation:

1) Sentence splitting

There is 3 sentence in total in sent8.txt

```
Sentences splitting results:
["On Monday, September 17, 2018, John O'Malley promised his colleague Mary that he would put a replacement apple in the office fridge.", "O'Malley intended to share it with her on Tuesday at his desk and anticipated that the crunchy treat would delight them both.", 'But she was sick that day.']
```

2) Named Entity Module

I use the result of “main_stan.py” in named entity module for sent8.txt

```
Name entities:
{"O'Malley", 'Mary', "John O'Malley"}
```

3) POS tagging

I use the result of “main_stan.py” in part-of-speech module for sent8.txt

```
Part-of-speech tagging results:
[[('On', 'IN'), ('Monday', 'NNP'), (',', ','), ('September', 'NNP'), ('17', 'CD'), (',', ','), ('2018', 'CD'), (',', ','), ("John O'Malley", 'NNP'), ('promised', 'VBD'), ('his', 'PRP$'), ('colleague', 'NN'), ('Mary', 'NNP'), ('that', 'IN'), ('he', 'PRP'), ('would', 'MD'), ('put', 'VB'), ('a', 'DT'), ('replacement', 'NN'), ('apple', 'NN'), ('in', 'IN'), ('the', 'DT'), ('office', 'NN'), ('fridge', 'NN')],
[("O'Malley", 'NNP'), ('intended', 'VBD'), ('to', 'TO'), ('share', 'VB'), ('it', 'PRP'), ('with', 'IN'), ('her', 'PRP'), ('on', 'IN'), ('Tuesday', 'NNP'), ('at', 'IN'), ('his', 'PRP$'), ('desk', 'NN'), ('and', 'CC'), ('anticipated', 'VBD'), ('that', 'IN'), ('the', 'DT'), ('crunchy', 'JJ'), ('treat', 'NN'), ('would', 'MD'), ('delight', 'VB'), ('them', 'PRP'), ('both', 'DT')], [('But', 'CC'), ('she', 'PRP'), ('was', 'VBD'), ('sick', 'JJ'), ('that', 'DT'), ('day', 'NN')]]
```

4) Earley Parser parsing

Sentence 1: On Monday, September 17, 2018, John O'Malley promised his colleague Mary that he would put a replacement apple in the office fridge.

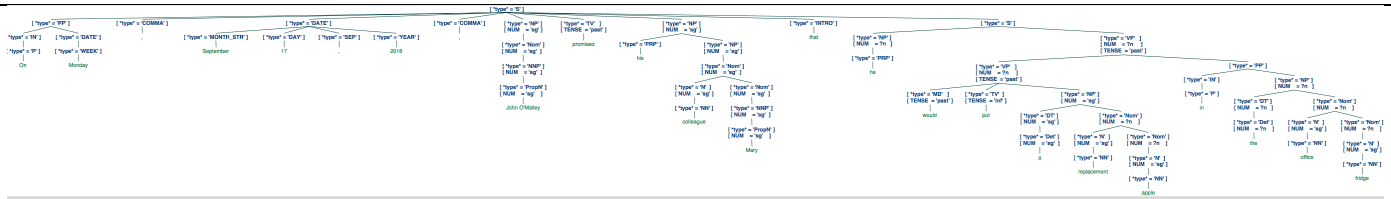
There are two parse trees generated for sentence 1.

In Parse tree 1, the PP constituent attaches to the verb “put”;

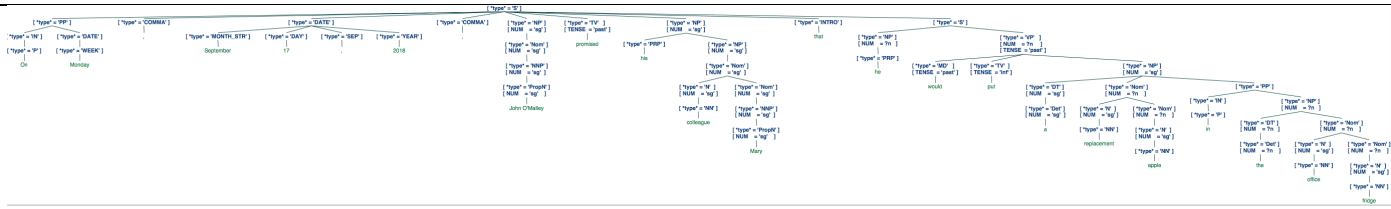
In Parse tree 2, the PP constituent modifies the NOM “apple”.

Based on the meaning, the parse tree 1 is the correct one.

Parse tree 1



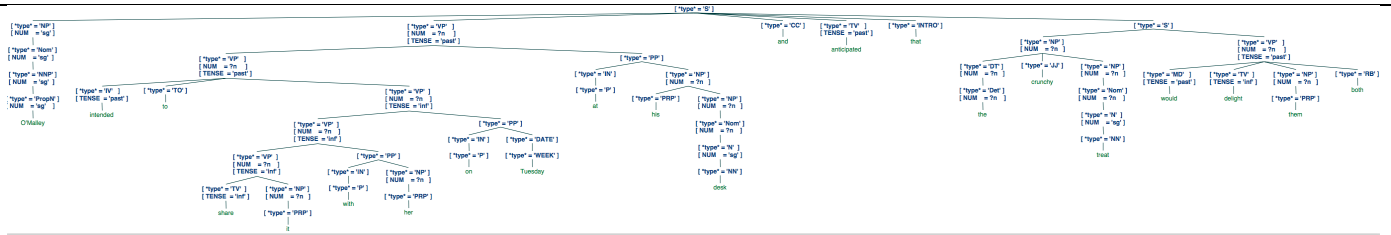
Parse tree 2



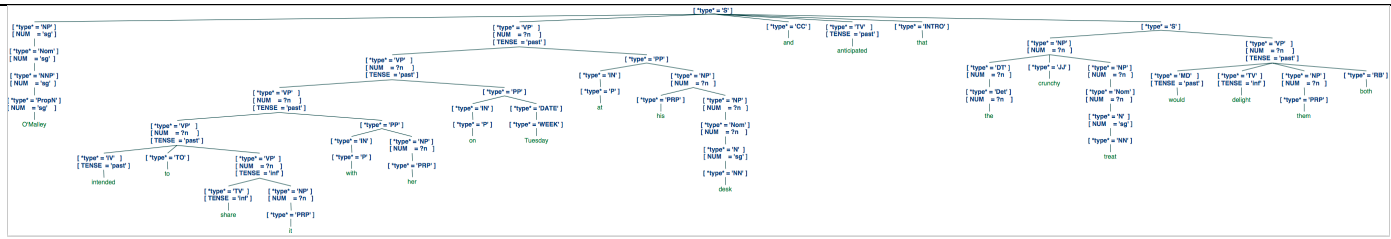
Sentence 2: O'Malley intended to share it with her on Tuesday at his desk and anticipated that the crunchy treat would delight them both.

There are 5 parse trees generated, since there are three PP constituents “with her”, “on Tuesday” and “at his desk”. **The correct parse tree is the parse tree 4** where “with her”, “on Tuesday” attach to “share”, and “at his desk” attaches to “it”.

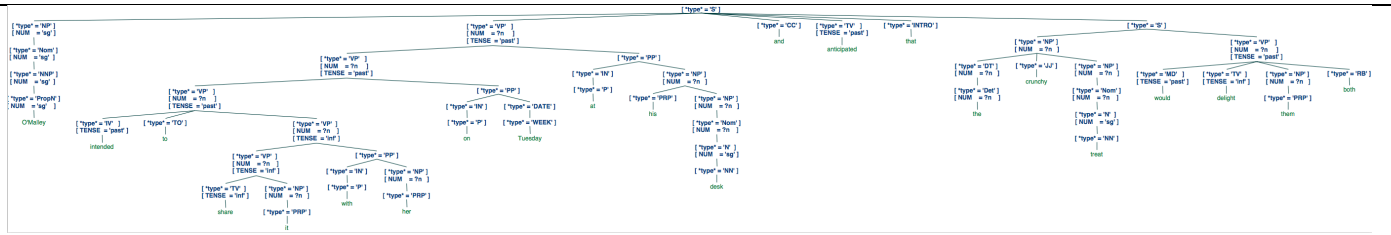
Parse tree 1

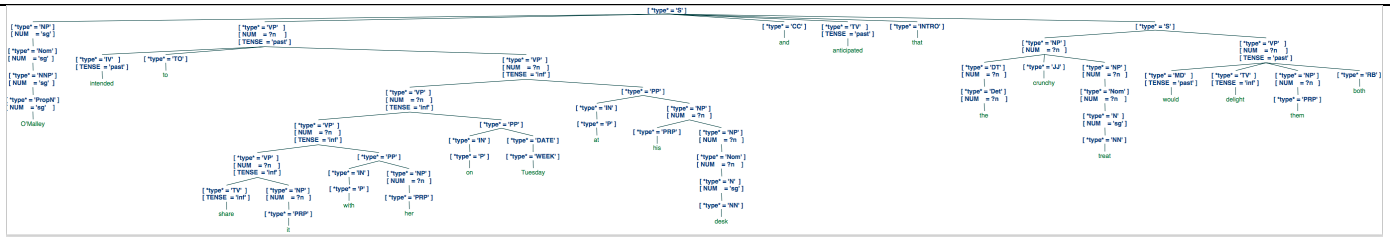
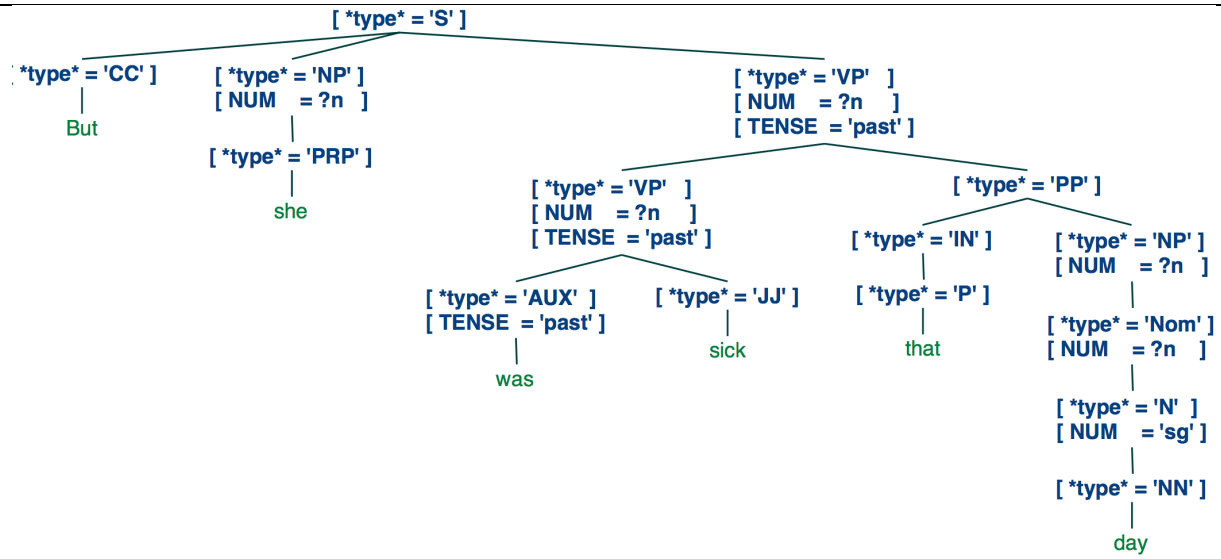


Parse tree 2



Parse tree 3



Parse tree 4	
Sentence 3: But she was sick that day.	
Parse tree	

9. Test case 9: sentence example 9

	Input file and content	Output on console and parse tree diagram	

Input file

data/sent9.txt

Console Output

Sentences splitting results:
["Sue said that on Monday, September 17, 2018, John O'Malley promised his colleague Mary that he would put a replacement apple in the office fridge and that O'Malley intended to share it with her on Tuesday at his desk."]

Part-of-speech tagging results:
[[('Sue', 'NNP'), ('said', 'VBD'), ('that', 'IN'), ('on', 'IN'), ('Monday', 'NNP'), (',', ','), ('September', 'NNP'), ('17', 'CD'), (',', ','), ('2018', 'CD'), (',', ','), ("John O'Malley", 'NNP'), ('promised', 'VBD'), ('his', 'PRP$'), ('colleague', 'NN'), ('Mary', 'NNP'), ('that', 'IN'), ('he', 'PRP'), ('would', 'MD'), ('put', 'VB'), ('a', 'DT'), ('replacement', 'NN'), ('apple', 'NN'), ('in', 'IN'), ('the', 'DT'), ('office', 'NN'), ('fridge', 'NN'), ('and', 'CC'), ('that', 'IN'), ("O'Malley", 'NNP'), ('intended', 'VBD'), ('to', 'TO'), ('share', 'VB'), ('it', 'PRP'), ('with', 'IN'), ('her', 'PRP'), ('on', 'IN'), ('Tuesday', 'NNP'), ('at', 'IN'), ('his', 'PRP$'), ('desk', 'NN')]]
Name entities:
{"John O'Malley", "O'Malley", 'Mary'}

Parsing results:

```
(S[]
 (NP[NUM='sg']
  (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Sue))))
 (TV[TENSE='past'] said)
 (INTRO[] that)
 (S[]
  (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Monday)))
  (COMMA[] ,)
  (DATE[]
   (MONTH_STR[] September)
   (DAY[] 17)
   (SEP[] ,)
   (YEAR[] 2018))
  (COMMA[] ,)
  (NP[NUM='sg']
   (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John O'Malley))))
  (TV[TENSE='past'] promised)
  (NP[NUM='sg']
   (PRP[] his)
   (NP[NUM='sg']
    (Nom[NUM='sg']
     (N[NUM='sg'] (NN[] colleague))
     (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary))))))
  (INTRO[] that)
  (S[]
   (NP[NUM=?n] (PRP[] he))
   (VP[NUM=?n, TENSE='past']
    (VP[NUM=?n, TENSE='past']
     (MD[TENSE='past'] would)
     (TV[TENSE='inf'] put)
     (NP[NUM='sg']
      (DT[NUM='sg'] (Det[NUM='sg'] a))
      (Nom[NUM=?n]
       (N[NUM='sg'] (NN[] replacement))
       (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))))
    (PP[]
     (IN[] (P[] in))
     (NP[NUM=?n]
      (DT[NUM=?n] (Det[NUM=?n] the))
      (Nom[NUM=?n]
       (N[NUM='sg'] (NN[] office))
       (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge)))))))))
 (CC[] and)
 (IN[] (P[] that))
 (S[]
  (NP[NUM='sg']
   (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] O'Malley))))
  (VP[NUM=?n, TENSE='past']
   (VP[NUM=?n, TENSE='past']
    (IV[TENSE='past'] intended)
    (TO[] to)
    (VP[NUM=?n, TENSE='inf']
     (VP[NUM=?n, TENSE='inf']
      (VP[NUM=?n, TENSE='inf']
       (TV[TENSE='inf'] share)
       (NP[NUM=?n] (PRP[] it)))
      (PP[] (IN[] (P[] with)) (NP[NUM=?n] (PRP[] her))))
     (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Tuesday)))))
   (PP[]
    (IN[] (P[] at))
    (NP[NUM=?n]
     (PRP[] his)
```

```
      (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] desk)))))))))
(S[]
 (NP[NUM='sg']
  (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Sue))))
 (TV[TENSE='past'] said)
 (INTRO[] that)
 (S[]
  (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Monday)))
  (COMMA[] ,)
  (DATE[]
   (MONTH_STR[] September)
   (DAY[] 17)
   (SEP[] ,)
   (YEAR[] 2018))
  (COMMA[] ,)
  (NP[NUM='sg']
   (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John O'Malley))))
  (TV[TENSE='past'] promised)
  (NP[NUM='sg']
   (PRP[] his)
   (NP[NUM='sg']
    (Nom[NUM='sg']
     (N[NUM='sg'] (NN[] colleague))
     (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary))))))
  (INTRO[] that)
  (S[]
   (NP[NUM=?n] (PRP[] he))
   (VP[NUM=?n, TENSE='past']
    (VP[NUM=?n, TENSE='past']
     (MD[TENSE='past'] would)
     (TV[TENSE='inf'] put)
     (NP[NUM='sg']
      (DT[NUM='sg'] (Det[NUM='sg'] a))
      (Nom[NUM=?n]
       (N[NUM='sg'] (NN[] replacement))
       (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))))
    (PP[]
     (IN[] (P[] in))
     (NP[NUM=?n]
      (DT[NUM=?n] (Det[NUM=?n] the))
      (Nom[NUM=?n]
       (N[NUM='sg'] (NN[] office))
       (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge)))))))))
 (CC[] and)
 (IN[] (P[] that))
 (S[]
  (NP[NUM='sg']
   (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] O'Malley))))
  (VP[NUM=?n, TENSE='past']
   (VP[NUM=?n, TENSE='past']
    (VP[NUM=?n, TENSE='past']
     (VP[NUM=?n, TENSE='past']
      (IV[TENSE='past'] intended)
      (TO[] to)
      (VP[NUM=?n, TENSE='inf']
       (TV[TENSE='inf'] share)
       (NP[NUM=?n] (PRP[] it))))
     (PP[] (IN[] (P[] with)) (NP[NUM=?n] (PRP[] her))))
    (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Tuesday))))
   (PP[]
    (IN[] (P[] at))
    (NP[NUM=?n]
     (PRP[] his)
     (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] desk)))))))))
(S[]
 (NP[NUM='sg']
  (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Sue))))
 (TV[TENSE='past'] said)
 (INTRO[] that)
 (S[]
  (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Monday)))
  (COMMA[] ,)
  (DATE[]
   (MONTH_STR[] September)
   (DAY[] 17)
   (SEP[] ,)
   (YEAR[] 2018))
  (COMMA[] ,)
  (NP[NUM='sg']
   (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John O'Malley))))
  (TV[TENSE='past'] promised)
  (NP[NUM='sg']
```

```
    (PRP[] his)
    (NP[NUM='sg']
     (Nom[NUM='sg']
      (N[NUM='sg'] (NN[] colleague))
      (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary))))))
   (INTRO[] that)
   (S[]
    (NP[NUM=?n] (PRP[] he))
    (VP[NUM=?n, TENSE='past']
     (VP[NUM=?n, TENSE='past']
      (MD[TENSE='past'] would)
      (TV[TENSE='inf'] put)
      (NP[NUM='sg']
       (DT[NUM='sg'] (Det[NUM='sg'] a))
       (Nom[NUM=?n]
        (N[NUM='sg'] (NN[] replacement))
        (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))))
     (PP[]
      (IN[] (P[] in))
      (NP[NUM=?n]
       (DT[NUM=?n] (Det[NUM=?n] the))
       (Nom[NUM=?n]
        (N[NUM='sg'] (NN[] office))
        (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge)))))))))
   (CC[] and)
   (IN[] (P[] that))
   (S[]
    (NP[NUM='sg']
     (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] O'Malley))))
    (VP[NUM=?n, TENSE='past']
     (VP[NUM=?n, TENSE='past']
      (VP[NUM=?n, TENSE='past']
       (IV[TENSE='past'] intended)
       (TO[] to)
       (VP[NUM=?n, TENSE='inf']
        (VP[NUM=?n, TENSE='inf']
         (TV[TENSE='inf'] share)
         (NP[NUM=?n] (PRP[] it)))
        (PP[] (IN[] (P[] with)) (NP[NUM=?n] (PRP[] her)))))
      (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Tuesday))))
     (PP[]
      (IN[] (P[] at))
      (NP[NUM=?n]
       (PRP[] his)
       (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] desk)))))))))
(S[]
 (NP[NUM='sg']
  (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Sue))))
 (TV[TENSE='past'] said)
 (INTRO[] that)
 (S[]
  (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Monday)))
  (COMMA[] ,)
  (DATE[]
   (MONTH_STR[] September)
   (DAY[] 17)
   (SEP[] ,)
   (YEAR[] 2018))
  (COMMA[] ,)
  (NP[NUM='sg']
   (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John O'Malley))))
  (TV[TENSE='past'] promised)
  (NP[NUM='sg']
   (PRP[] his)
   (NP[NUM='sg']
    (Nom[NUM='sg']
     (N[NUM='sg'] (NN[] colleague))
     (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary))))))
  (INTRO[] that)
  (S[]
   (NP[NUM=?n] (PRP[] he))
   (VP[NUM=?n, TENSE='past']
    (VP[NUM=?n, TENSE='past']
     (MD[TENSE='past'] would)
     (TV[TENSE='inf'] put)
     (NP[NUM='sg']
```

```
          (DT[NUM='sg'] (Det[NUM='sg'] a))
          (Nom[NUM=?n]
           (N[NUM='sg'] (NN[] replacement))
           (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))))
       (PP[]
        (IN[] (P[] in))
        (NP[NUM=?n]
         (DT[NUM=?n] (Det[NUM=?n] the))
         (Nom[NUM=?n]
          (N[NUM='sg'] (NN[] office))
          (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge)))))))))
  (CC[] and)
  (IN[] (P[] that))
  (S[]
   (NP[NUM='sg']
    (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] O'Malley))))
   (VP[NUM=?n, TENSE='past']
    (IV[TENSE='past'] intended)
    (TO[] to)
    (VP[NUM=?n, TENSE='inf']
     (VP[NUM=?n, TENSE='inf']
      (VP[NUM=?n, TENSE='inf']
       (VP[NUM=?n, TENSE='inf']
        (TV[TENSE='inf'] share)
        (NP[NUM=?n] (PRP[] it)))
       (PP[] (IN[] (P[] with)) (NP[NUM=?n] (PRP[] her))))
      (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Tuesday))))
     (PP[]
      (IN[] (P[] at))
      (NP[NUM=?n]
       (PRP[] his)
       (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] desk)))))))))
(S[]
 (NP[NUM='sg']
  (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Sue))))
 (TV[TENSE='past'] said)
 (INTRO[] that)
 (S[]
  (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Monday)))
  (COMMA[] ,)
  (DATE[]
   (MONTH_STR[] September)
   (DAY[] 17)
   (SEP[] ,)
   (YEAR[] 2018))
  (COMMA[] ,)
  (NP[NUM='sg']
   (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John O'Malley))))
  (TV[TENSE='past'] promised)
  (NP[NUM='sg']
   (PRP[] his)
   (NP[NUM='sg']
    (Nom[NUM='sg']
     (N[NUM='sg'] (NN[] colleague))
     (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary))))))
  (INTRO[] that)
  (S[]
   (NP[NUM=?n] (PRP[] he))
   (VP[NUM=?n, TENSE='past']
    (MD[TENSE='past'] would)
    (TV[TENSE='inf'] put)
    (NP[NUM='sg']
     (DT[NUM='sg'] (Det[NUM='sg'] a))
     (Nom[NUM=?n]
      (N[NUM='sg'] (NN[] replacement))
      (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))
     (PP[]
      (IN[] (P[] in))
      (NP[NUM=?n]
       (DT[NUM=?n] (Det[NUM=?n] the))
       (Nom[NUM=?n]
        (N[NUM='sg'] (NN[] office))
        (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge)))))))))
  (CC[] and)
  (IN[] (P[] that))
  (S[]
```

```
    (NP[NUM='sg']
      (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] O'Malley))))
    (VP[NUM=?n, TENSE='past']
      (VP[NUM=?n, TENSE='past']
        (IV[TENSE='past'] intended)
        (TO[] to)
        (VP[NUM=?n, TENSE='inf']
          (VP[NUM=?n, TENSE='inf']
            (VP[NUM=?n, TENSE='inf']
              (TV[TENSE='inf'] share)
              (NP[NUM=?n] (PRP[] it)))
            (PP[] (IN[] (P[] with)) (NP[NUM=?n] (PRP[] her))))
          (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Tuesday)))))
      (PP[]
        (IN[] (P[] at))
        (NP[NUM=?n]
          (PRP[] his)
          (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] desk)))))))))
  (S[]
    (NP[NUM='sg']
      (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Sue))))
    (TV[TENSE='past'] said)
    (INTRO[] that)
    (S[]
      (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Monday)))
      (COMMA[] ,)
      (DATE[]
        (MONTH_STR[] September)
        (DAY[] 17)
        (SEP[] ,)
        (YEAR[] 2018))
      (COMMA[] ,)
      (NP[NUM='sg']
        (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John O'Malley))))
      (TV[TENSE='past'] promised)
      (NP[NUM='sg']
        (PRP[] his)
        (NP[NUM='sg']
          (Nom[NUM='sg']
            (N[NUM='sg'] (NN[] colleague))
            (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary))))))
      (INTRO[] that)
      (S[]
        (NP[NUM=?n] (PRP[] he))
        (VP[NUM=?n, TENSE='past']
          (MD[TENSE='past'] would)
          (TV[TENSE='inf'] put)
          (NP[NUM='sg']
            (DT[NUM='sg'] (Det[NUM='sg'] a))
            (Nom[NUM=?n]
              (N[NUM='sg'] (NN[] replacement))
              (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))
            (PP[]
              (IN[] (P[] in))
              (NP[NUM=?n]
                (DT[NUM=?n] (Det[NUM=?n] the))
                (Nom[NUM=?n]
                  (N[NUM='sg'] (NN[] office))
                  (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge))))))))))
      (CC[] and)
      (IN[] (P[] that))
      (S[]
        (NP[NUM='sg']
          (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] O'Malley))))
        (VP[NUM=?n, TENSE='past']
          (VP[NUM=?n, TENSE='past']
            (VP[NUM=?n, TENSE='past']
              (VP[NUM=?n, TENSE='past']
                (IV[TENSE='past'] intended)
                (TO[] to)
                (VP[NUM=?n, TENSE='inf']
                  (TV[TENSE='inf'] share)
                  (NP[NUM=?n] (PRP[] it))))
              (PP[] (IN[] (P[] with)) (NP[NUM=?n] (PRP[] her))))
            (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Tuesday))))
          (PP[]
            (IN[] (P[] at))
            (NP[NUM=?n]
              (PRP[] his)
              (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] desk)))))))))
  (S[]
    (NP[NUM='sg']
```

```
  (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Sue))))
 (TV[TENSE='past'] said)
 (INTRO[] that)
 (S[]
  (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Monday)))
  (COMMA[] ,)
  (DATE[]
   (MONTH_STR[] September)
   (DAY[] 17)
   (SEP[] ,)
   (YEAR[] 2018))
  (COMMA[] ,)
  (NP[NUM='sg']
   (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John O'Malley))))
  (TV[TENSE='past'] promised)
  (NP[NUM='sg']
   (PRP[] his)
   (NP[NUM='sg']
    (Nom[NUM='sg']
     (N[NUM='sg'] (NN[] colleague))
     (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary))))))
  (INTRO[] that)
  (S[]
   (NP[NUM=?n] (PRP[] he))
   (VP[NUM=?n, TENSE='past']
    (MD[TENSE='past'] would)
    (TV[TENSE='inf'] put)
    (NP[NUM='sg']
     (DT[NUM='sg'] (Det[NUM='sg'] a))
     (Nom[NUM=?n]
      (N[NUM='sg'] (NN[] replacement))
      (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))
     (PP[]
      (IN[] (P[] in))
      (NP[NUM=?n]
       (DT[NUM=?n] (Det[NUM=?n] the))
       (Nom[NUM=?n]
        (N[NUM='sg'] (NN[] office))
        (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge)))))))))
  (CC[] and)
  (IN[] (P[] that))
  (S[]
   (NP[NUM='sg']
    (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] O'Malley))))
   (VP[NUM=?n, TENSE='past']
    (VP[NUM=?n, TENSE='past']
     (VP[NUM=?n, TENSE='past']
      (IV[TENSE='past'] intended)
      (TO[] to)
      (VP[NUM=?n, TENSE='inf']
       (VP[NUM=?n, TENSE='inf']
        (TV[TENSE='inf'] share)
        (NP[NUM=?n] (PRP[] it)))
       (PP[] (IN[] (P[] with)) (NP[NUM=?n] (PRP[] her)))))
     (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Tuesday))))
    (PP[]
     (IN[] (P[] at))
     (NP[NUM=?n]
      (PRP[] his)
      (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] desk)))))))))
(S[]
 (NP[NUM='sg']
  (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Sue))))
 (TV[TENSE='past'] said)
 (INTRO[] that)
 (S[]
  (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Monday)))
  (COMMA[] ,)
  (DATE[]
   (MONTH_STR[] September)
   (DAY[] 17)
   (SEP[] ,)
   (YEAR[] 2018))
  (COMMA[] ,)
  (NP[NUM='sg']
   (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John O'Malley))))
  (TV[TENSE='past'] promised)
  (NP[NUM='sg']
   (PRP[] his)
   (NP[NUM='sg']
    (Nom[NUM='sg']
     (N[NUM='sg'] (NN[] colleague))
```

		(Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary)))))) (INTRO[] that) (S[] (NP[NUM=?n] (PRP[] he)) (VP[NUM=?n, TENSE='past'] (MD[TENSE='past'] would) (TV[TENSE='inf'] put) (NP[NUM='sg'] (DT[NUM='sg'] (Det[NUM='sg'] a)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] replacement)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple)))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] office)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge))))))))) (CC[] and) (IN[] (P[] that)) (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] O'Malley)))) (VP[NUM=?n, TENSE='past'] (IV[TENSE='past'] intended) (TO[] to) (VP[NUM=?n, TENSE='inf'] (VP[NUM=?n, TENSE='inf'] (VP[NUM=?n, TENSE='inf'] (VP[NUM=?n, TENSE='inf'] (TV[TENSE='inf'] share) (NP[NUM=?n] (PRP[] it))) (PP[] (IN[] (P[] with)) (NP[NUM=?n] (PRP[] her)))) (PP[] (IN[] (P[] on)) (DATE[] (WEEK[] Tuesday)))) (PP[] (IN[] (P[] at)) (NP[NUM=?n] (PRP[] his) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] desk))))))))))	
content	Sue said that on Monday, September 17, 2018, John O'Malley promised his colleague Mary that he would put a replacement apple in the office fridge and that O'Malley intended to share it with her on Tuesday at his desk.	Since there are 8 parse trees for this sentence and the diagrams are really large, I saved the parse trees I got in the directory “code/got_results/sent9_diagrams/”. So for saving spaces, I only attached the correct parse tree in this Demo report. *Tree4_diagram.png* 	Parse Tree Diagram

Test case 9 result explanation:

1) Sentence splitting

There is 1 sentence in total in sent9.txt.

```
Sentences splitting results:
["Sue said that on Monday, September 17, 2018, John O'Malley promised his colleague Mary that he would put a replacement apple in the office fridge and that O'Malley intended to share it with her on Tuesday at his desk."]
```

2) Named Entity Module

I use the union result of “main_stan.py” and "main_base" in named entity module for sent9.txt.

```
Name entities:
{‘Sue’, "John O'Malley", "O'Malley", 'Mary'}
```

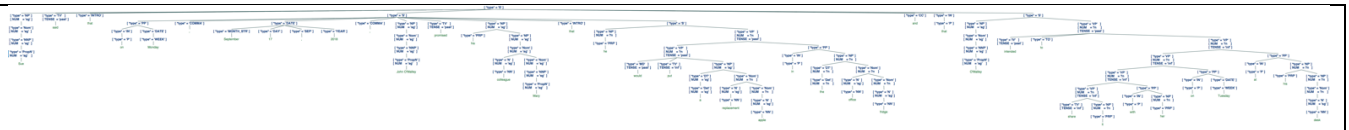
3) POS tagging

I use the result of “main_stan.py” in part-of-speech module for sent9.txt.

```
Part-of-speech tagging results:
[[('Sue', 'NNP'), ('said', 'VBD'), ('that', 'IN'), ('on', 'IN'), ('Monday', 'NNP'), (',', ','), ('September', 'NNP'), ('17', 'CD'), (',', ','),
('2018', 'CD'), (',', ','), ("John O'Malley", 'NNP'), ('promised', 'VBD'), ('his', 'PRP$'), ('colleague', 'NN'), ('Mary', 'NNP'),
('that', 'IN'), ('he', 'PRP'), ('would', 'MD'), ('put', 'VB'), ('a', 'DT'), ('replacement', 'NN'), ('apple', 'NN'), ('in', 'IN'), ('the', 'DT'),
('office', 'NN'), ('fridge', 'NN'), ('and', 'CC'), ('that', 'IN'), ("O'Malley", 'NNP'), ('intended', 'VBD'), ('to', 'TO'), ('share', 'VB'),
('it', 'PRP'), ('with', 'IN'), ('her', 'PRP'), ('on', 'IN'), ('Tuesday', 'NNP'), ('at', 'IN'), ('his', 'PRP$'), ('desk', 'NN')]]
```

4) Earley Parser parsing

Due to the three PP constituent “in the office fridge” and “at his desk” and “with her”, more possible parse trees are generated. **The parse tree 4 is the correct tree** for this sentence.

parse tree 4	

10.Test case 10: challenge text 1

	Input file and content	Output on console and parse tree diagram	
Input file	data/challenge1.txt	Sentences splitting results: ['John ate at his desk.'] --- Part-of-speech tagging results: [[('John', 'NNP'), ('ate', 'VBD'), ('at', 'IN'), ('his', 'PRP$'), ('desk', 'NN')]] Name entities: {'John'} --- Parsing results: (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (VP[NUM=?n, TENSE='past'] (VP[NUM=?n, TENSE='past'] (IV[TENSE='past'] ate)) (PP[] (IN[] (P[] at)) (NP[NUM=?n] (PRP[] his) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] desk))))))))	Console Output
content	John ate at his desk.	There is only 1 parse tree generated, please see the diagram below.	Parse Tree Diagram

Challenge text 1 result explanation:

1) Sentence splitting

There is 1 sentence in total in challenge.txt, and the result matches

```
Sentences splitting results:
['John ate at his desk.']
```

2) Named Entity Module

I use the union result of “main_stan.py” in named entity module for challenge1.txt.

```
Name entities:
{'John'}
```

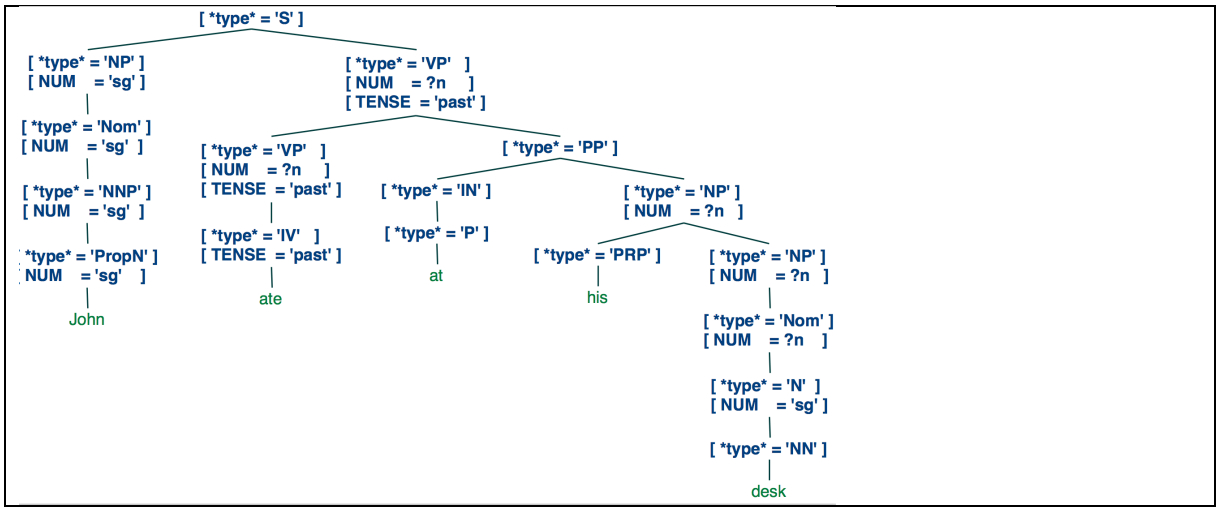
3) POS tagging

I use the result of “main_stan.py” in part-of-speech module for challenge1.txt

```
Part-of-speech tagging results:
[[('John', 'NNP'), ('ate', 'VBD'), ('at', 'IN'), ('his', 'PRP$'), ('desk', 'NN')]]
```

4) Earley Parser parsing

There is only 1 parse tree generated.



11.Test case 11: challenge text 2

	Input file and content	Output on console and parse tree diagram	
Input file	data/challenge2.txt	Sentences splitting results: ["Finally, O'Malley ate the crunchy apple on the table."] --- Part-of-speech tagging results: [[('Finally', 'RB'), (',', ','), ("O'Malley", 'NNP'), ('ate', 'VBD'), ('the', 'DT'), ('crunchy', 'JJ'), ('apple', 'NN'), ('on', 'IN'), ('the', 'DT'), ('table', 'NN')]] Name entities: {"O'Malley"} --- Parsing results: (S[] (RB[] Finally) (COMMA[] ,) (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] O'Malley)))) (VP[NUM=?n, TENSE='past'] (VP[NUM=?n, TENSE='past'] (TV[TENSE='past'] ate) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (JJ[] crunchy) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple)))))) (PP[] (IN[] (P[] on)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] table))))))) (S[] (RB[] Finally) (COMMA[] ,) (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] O'Malley)))) (VP[NUM=?n, TENSE='past'] (TV[TENSE='past'] ate) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (JJ[] crunchy) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))) (PP[] (IN[] (P[] on)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] table)))))))))	Console Output

content	Finally, O'Malley ate the crunchy apple on the table.	There are 2 parse trees, please see the diagrams below.	Parse Tree Diagram

Challenge text 2 result explanation:

5) Sentence splitting

There is 1 sentence in total in challenge.txt, and the result matches.

```
Sentences splitting results:
["Finally, O'Malley ate the crunchy apple on the table."]
```

6) Named Entity Module

I use the union result of “main_stan.py” in named entity module for challenge2.txt.

```
Name entities:
{"O'Malley"}
```

7) POS tagging

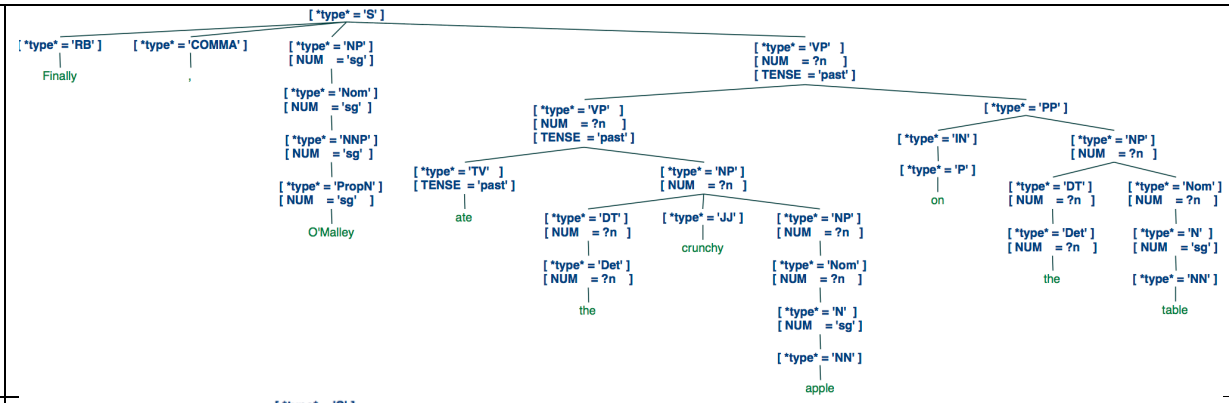
I use the result of “main_stan.py” in part-of-speech module for challenge2.txt.

```
Part-of-speech tagging results:
[[('Finally', 'RB'), (',', ','), ("O'Malley", 'NNP'), ('ate', 'VBD'), ('the', 'DT'), ('crunchy', 'JJ'), ('apple', 'NN'), ('on', 'IN'), ('the', 'DT'), ('table', 'NN')]]
```

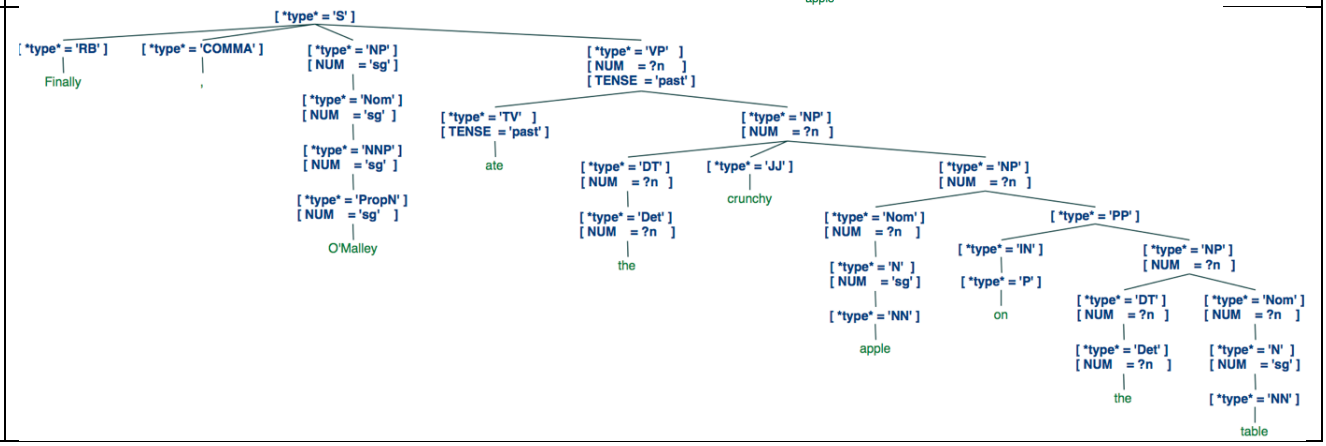
8) Earley Parser parsing

There are 2 parse trees for this sentence, however, the PP constituent “on the table” should attach to the NOM “table”, **so the parse tree 2 is correct.**

Parse tree 1



Parse tree 2



12.Test case 12: challenge text 3

	Input file and content	Output on console and parse tree diagram	
Input file	data/challenge3.txt	Sentences splitting results: ['Sue intended to share the fridge with John and Mary.'] --- Part-of-speech tagging results: [[('Sue', 'NNP'), ('intended', 'VBD'), ('to', 'TO'), ('share', 'VB'), ('the', 'DT'), ('fridge', 'NN'), ('with', 'IN'), ('John', 'NNP'), ('and', 'CC'), ('Mary', 'NNP')]] Name entities: {'Mary', 'John'} --- Parsing results: (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Sue)))) (VP[NUM=?n, TENSE='past'] (VP[NUM=?n, TENSE='past'] (IV[TENSE='past'] intended) (TO[] to) (VP[NUM=?n, TENSE='inf'] (TV[TENSE='inf'] share) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge)))))) (PP[] (IN[] (P[] with)) (NP[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)) (CC[] and) (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary)))))))) (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Sue)))) (VP[NUM=?n, TENSE='past'] (IV[TENSE='past'] intended) (TO[] to) (VP[NUM=?n, TENSE='inf'] (VP[NUM=?n, TENSE='inf'] (TV[TENSE='inf'] share) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge))))) (PP[] (IN[] (P[] with)) (NP[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)) (CC[] and) (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary))))))))) (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Sue)))) (VP[NUM=?n, TENSE='past'] (IV[TENSE='past'] intended) (TO[] to) (VP[NUM=?n, TENSE='inf'] (TV[TENSE='inf'] share) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge))) (PP[] (IN[] (P[] with)) (NP[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)) (CC[] and) (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary))))))))))	Console Output

content	Sue intended to share the fridge with John and Mary.	There are 3 parse trees generated, please see the diagrams below.	Parse Tree Diagram

Challenge text 3 result explanation:

9) Sentence splitting

There is 1 sentence in total in challenge3.txt, and the result matches.

```
Sentences splitting results:
['Sue intended to share the fridge with John and Mary.']
```

10) Named Entity Module

I use the union result of “main_stan.py” in named entity module for challenge3.txt.

```
Name entities:
{'Mary', 'John'}
```

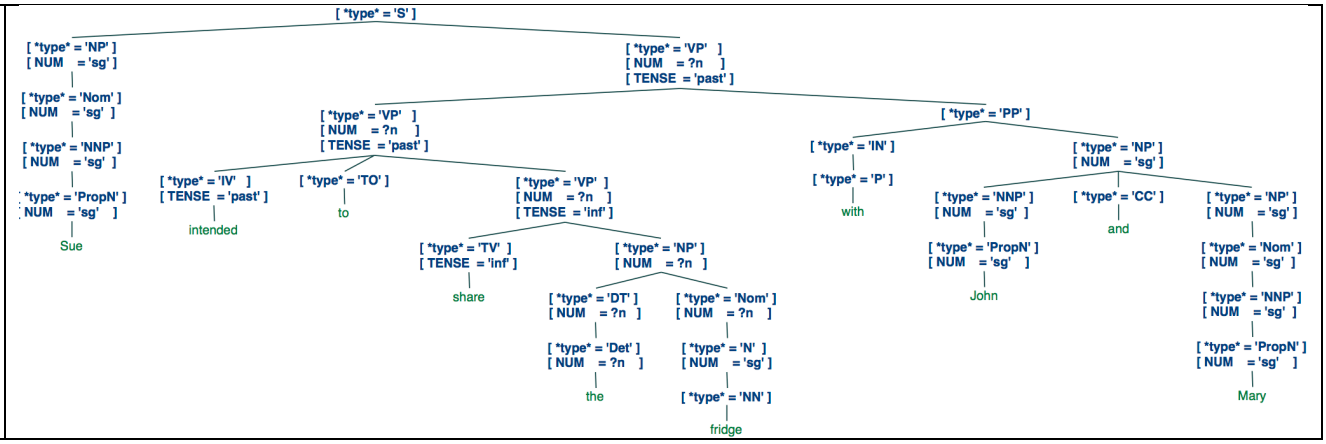
11) POS tagging

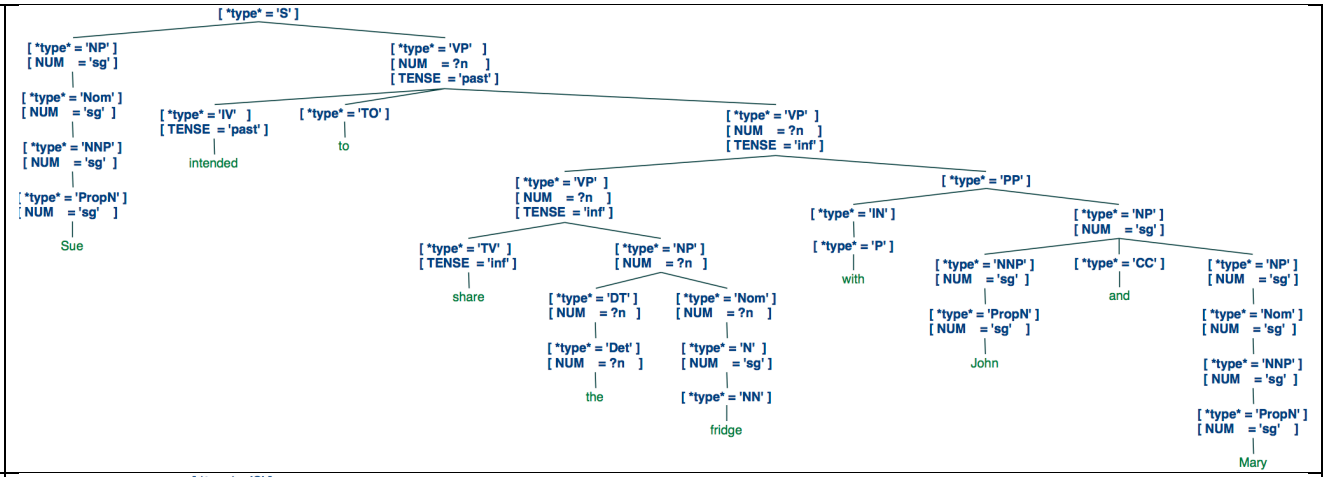
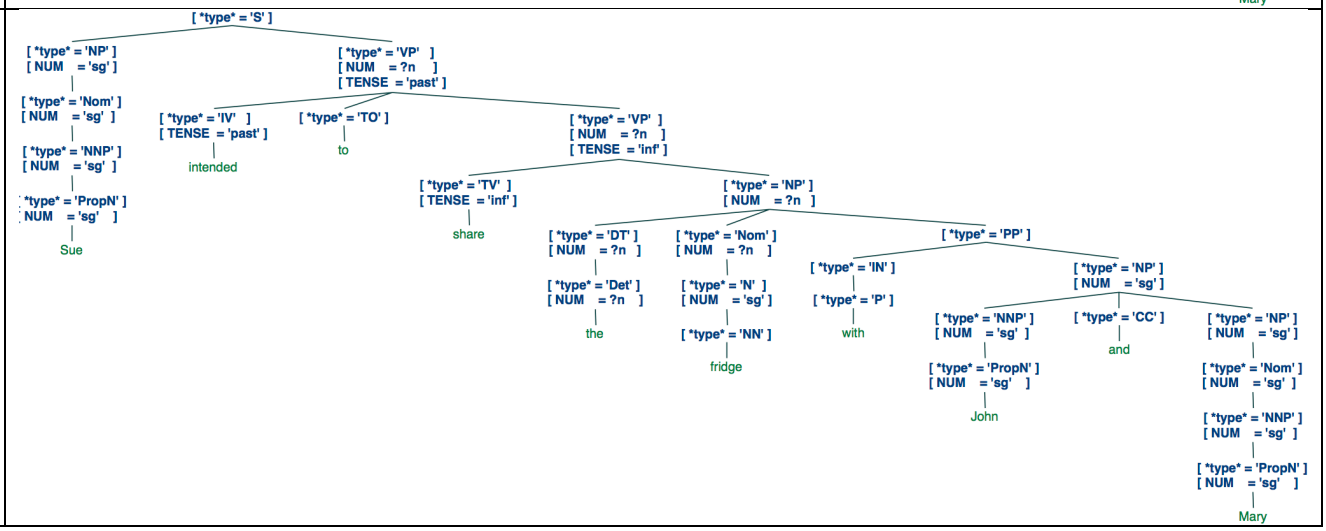
I use the result of “main_stan.py” in part-of-speech module for challenge3.txt.

```
Part-of-speech tagging results:
[[('Sue', 'NNP'), ('intended', 'VBD'), ('to', 'TO'), ('share', 'VB'), ('the', 'DT'), ('fridge', 'NN'), ('with', 'IN'), ('John', 'NNP'),
('and', 'CC'), ('Mary', 'NNP')]]
```

12) Earley Parser parsing

Parse tree 1



Parse tree 2	
Parse tree 3	

13.Test case 13: challenge text 4

	Input file and content	Output on console and parse tree diagram	
Input file	data/challenge4.txt	Sentences splitting results: ['Mary put apples in the fridge last Monday.'] --- Part-of-speech tagging results: [[('Mary', 'NNP'), ('put', 'VBD'), ('apples', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('fridge', 'NN'), ('last', 'JJ'), ('Monday', 'NNP')]] Name entities: {'Mary'} --- Parsing results: (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary)))) (VP[NUM=?n, TENSE='past'] (VP[NUM=?n, TENSE='past'] (TV[TENSE='past'] put) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='pl'] (NNS[] apples))))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge)))))) (DATE[] (JJ[] last) (WEEK[] Monday))) (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary)))) (VP[NUM=?n, TENSE='inf'] (VP[NUM=?n, TENSE='inf'] (TV[TENSE='inf'] put) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='pl'] (NNS[] apples))))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge)))))) (DATE[] (JJ[] last) (WEEK[] Monday))) (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary)))) (VP[NUM=?n, TENSE='inf'] (TV[TENSE='inf'] put) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='pl'] (NNS[] apples))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge))))))) (DATE[] (JJ[] last) (WEEK[] Monday))) (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] Mary)))) (VP[NUM=?n, TENSE='past'] (TV[TENSE='past'] put) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='pl'] (NNS[] apples))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge))))))) (DATE[] (JJ[] last) (WEEK[] Monday)))	Console Output

content	Mary put apples in the fridge last Monday.	There are 4 parse trees generated, please see the diagrams below.	Parse Tree Diagram

Challenge text 4 result explanation:

13) Sentence splitting

There is 1 sentence in total in challenge4.txt, and the result matches.

```
Sentences splitting results:
['Mary put apples in the fridge last Monday.']
```

14) Named Entity Module

I use the union result of “main_stan.py” in named entity module for challenge4.txt.

```
Name entities:
{'Mary'}
```

15) POS tagging

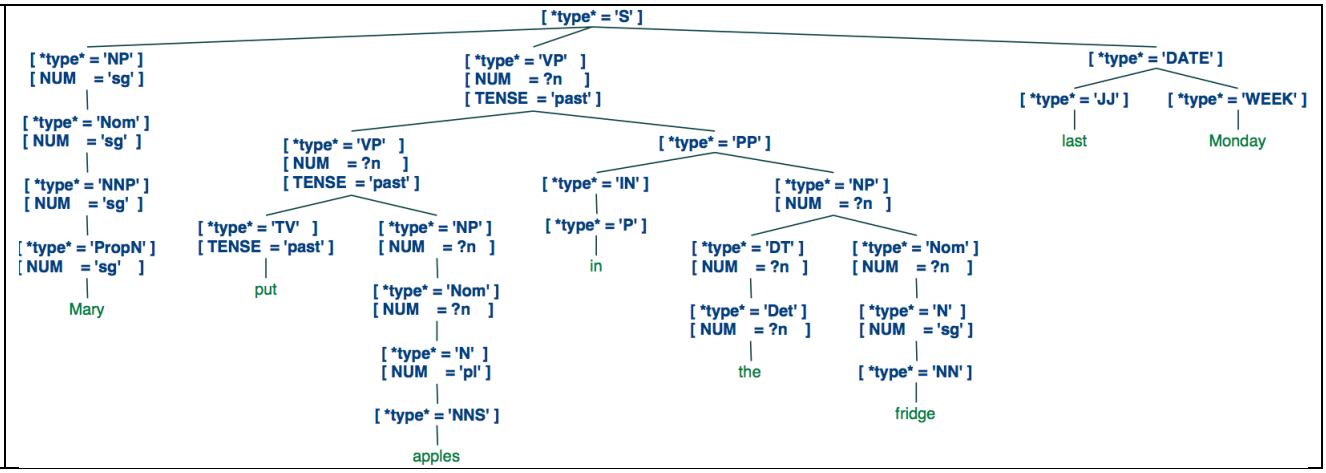
I use the result of “main_stan.py” in part-of-speech module for challenge4.txt.

```
Part-of-speech tagging results:
[[('Mary', 'NNP'), ('put', 'VBD'), ('apples', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('fridge', 'NN'), ('last', 'JJ'), ('Monday', 'NNP')]]
```

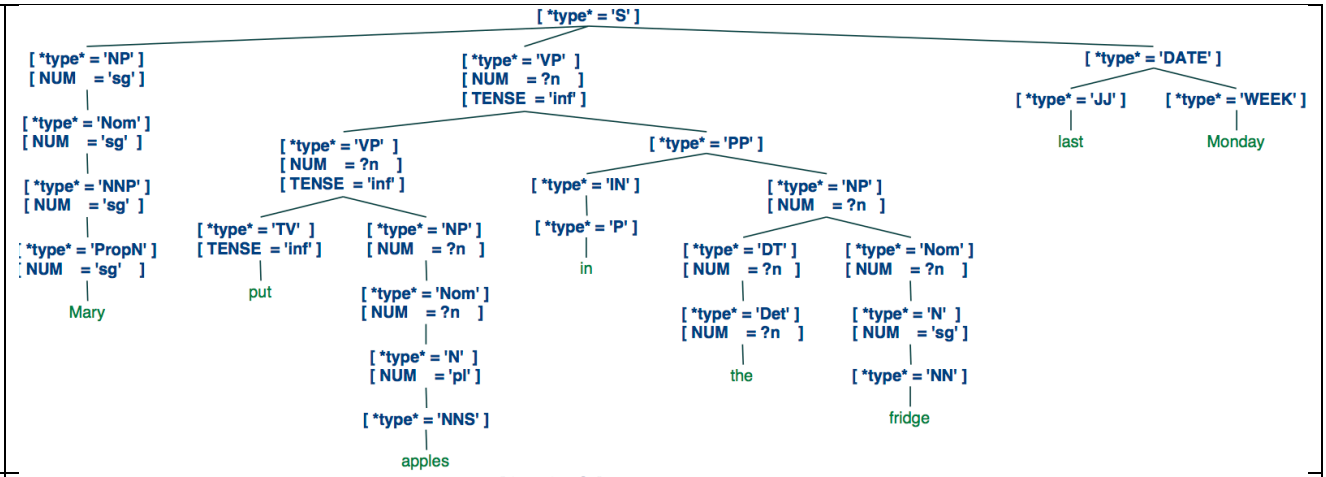
16) Earley Parser parsing

Parse tree is the correct one.

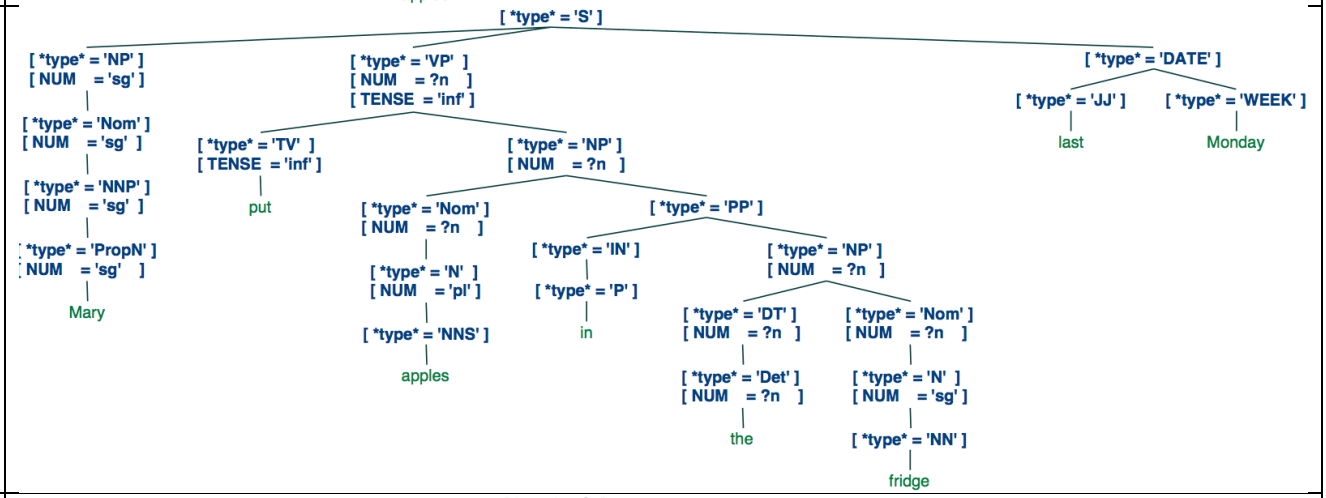
Parse tree 1



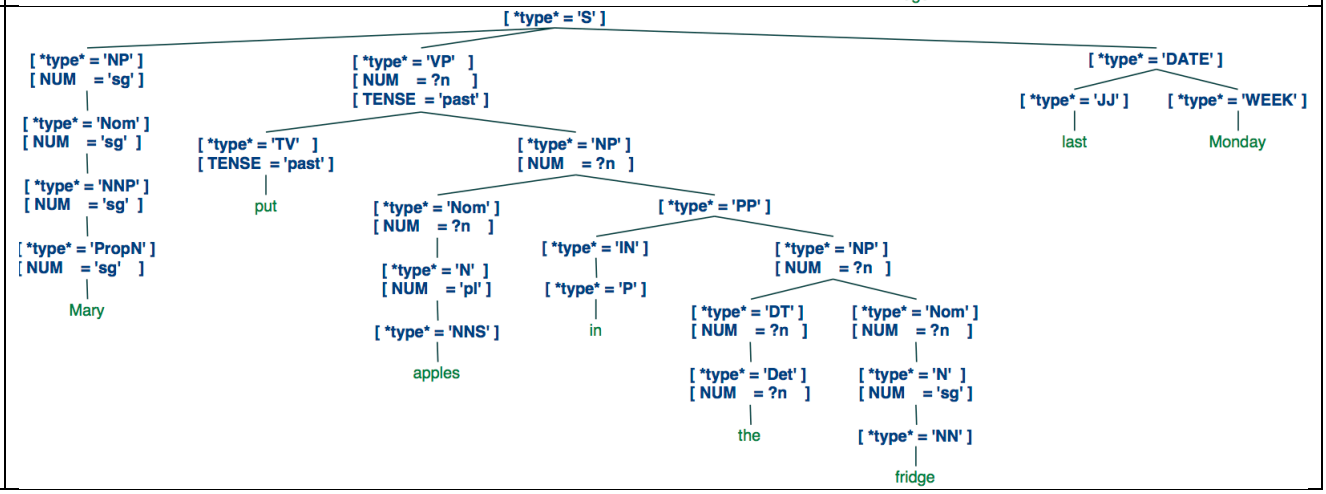
Parse tree 2



Parse tree 3



Parse tree 4



14.Test case 14: challenge text 5

	Input file and content	Output on console and parse tree diagram	
Input file	data/challenge5.txt	Sentences splitting results: ['I promise you an apple for September 2021.'] --- Part-of-speech tagging results: [[('I', 'PRP'), ('promise', 'VBP'), ('you', 'PRP'), ('an', 'DT'), ('apple', 'NN'), ('for', 'IN'), ('September', 'NNP'), ('2021', 'CD')]] Name entities: set() --- Parsing results: (S[] (NP[NUM=?n] (PRP[] I)) (VP[NUM=?n, TENSE='inf'] (VP[NUM=?n, TENSE='inf'] (TV[TENSE='inf'] promise) (NP[NUM='sg'] (PRP[] you) (NP[NUM='sg'] (DT[NUM='sg'] (Det[NUM='sg'] an)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple)))))) (PP[] (IN[] (P[] for)) (DATE[] (MONTH_STR[] September) (YEAR[] 2021))))) (S[] (NP[NUM=?n] (PRP[] I)) (VP[NUM=?n, TENSE='inf'] (TV[TENSE='inf'] promise) (NP[NUM='sg'] (PRP[] you) (NP[NUM='sg'] (DT[NUM='sg'] (Det[NUM='sg'] an)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))) (PP[] (IN[] (P[] for)) (DATE[] (MONTH_STR[] September) (YEAR[] 2021)))))))	Console Output
content	I promise you an apple for September 2021.	There are 2 parse trees generated, please see the diagrams below.	Parse Tree Diagram

Challenge text 5 result explanation:

17) Sentence splitting

There is 1 sentence in total in challenge5.txt, and the result matches.

```
Sentences splitting results:
['I promise you an apple for September 2021.']
```

18) Named Entity Module

I use the union result of “main_stan.py” in named entity module for challenge5.txt, and there is no named entities in the sentence.

```
Name entities:
set()
```

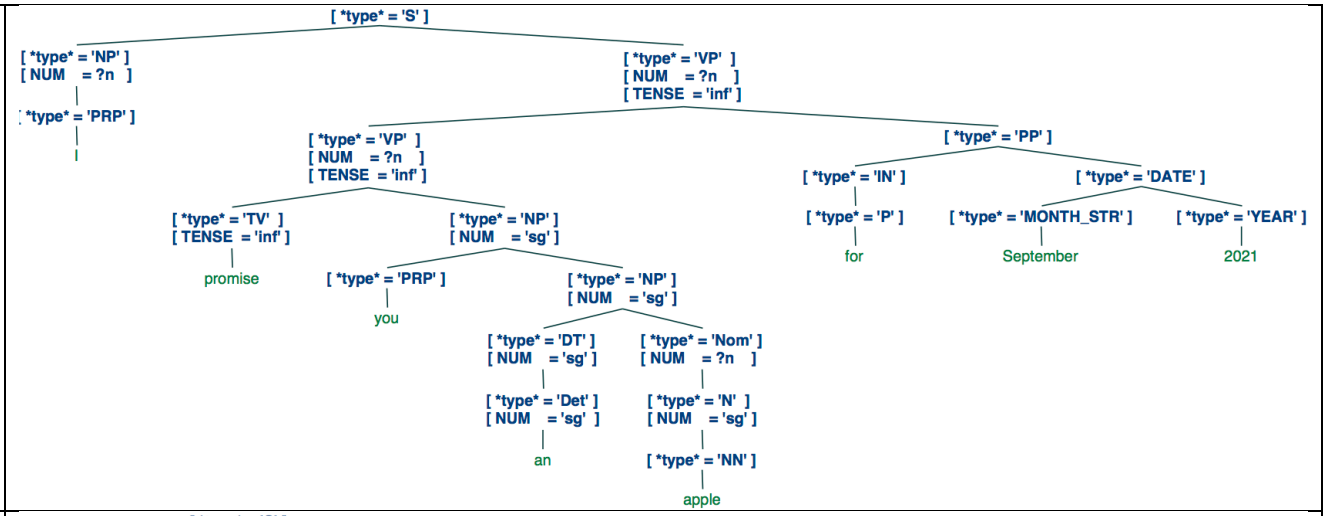
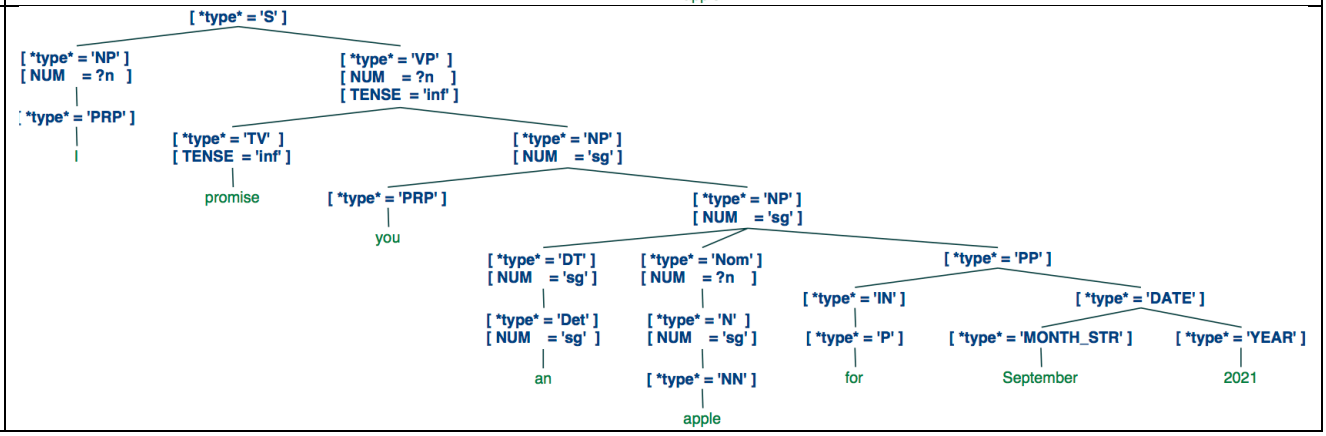
19) POS tagging

I use the result of “main_stan.py” in part-of-speech module for challenge.txt

```
Part-of-speech tagging results:
[[('I', 'PRP'), ('promise', 'VBP'), ('you', 'PRP'), ('an', 'DT'), ('apple', 'NN'), ('for', 'IN'), ('September', 'NNP'), ('2021', 'CD')]]
```

20) Earley Parser parsing

The Parse tree 1 is the correct one.

Parse tree 1	
Parse tree 2	

15.Test case 15: challenge text 6

	Input file and content	Output on console and parse tree diagram	
Input file	data/challenge6.txt	Sentences splitting results: ['It will be in the office fridge.'] --- Part-of-speech tagging results: [[('It', 'PRP'), ('will', 'MD'), ('be', 'VB'), ('in', 'IN'), ('the', 'DT'), ('office', 'NN'), ('fridge', 'NN')]] Name entities: set() --- Parsing results: (S[] (NP[NUM=?n] (PRP[] It)) (VP[NUM=?n, TENSE='inf'] (VP[NUM=?n, TENSE='inf'] (MD[TENSE='inf'] will) (AUX[TENSE='inf'] be)) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] office)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] fridge))))))))	Console Output
content	It will be in the office fridge.	There is only 1 parse tree generated, please see the parse tree below.	Parse Tree Diagram

Challenge text 6 result explanation:

21) Sentence splitting

There is 1 sentence in total in challenge6.txt, and the result matches.

```
Sentences splitting results:
['It will be in the office fridge.']
```

22) Named Entity Module

I use the union result of “main_stan.py” in named entity module for challenge6.txt, and there is no named entities in the sentence.

```
Name entities:
set()
```

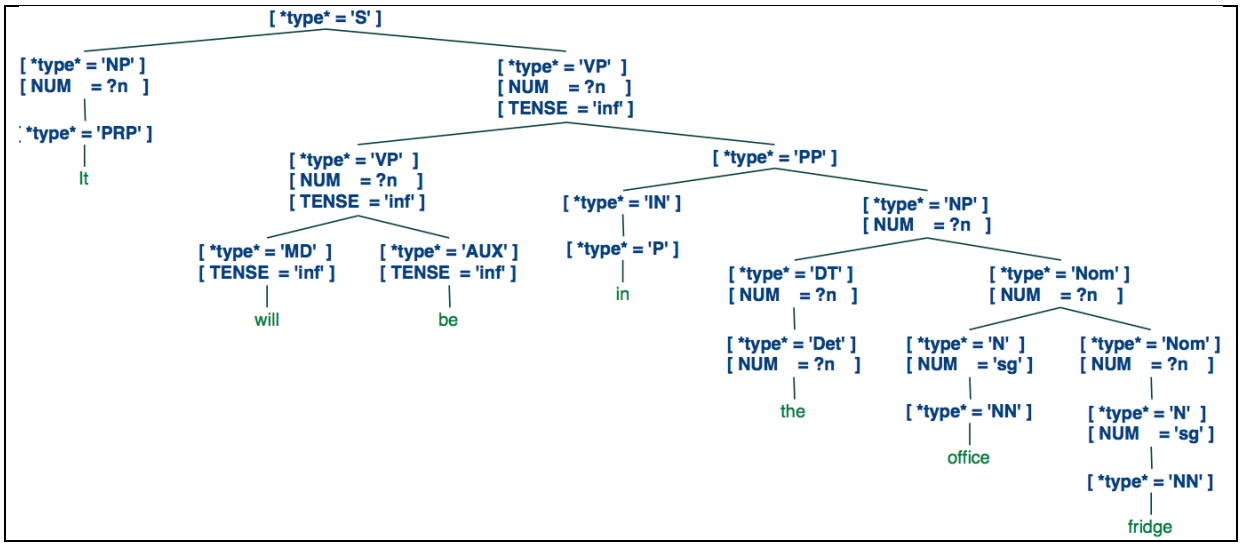
23) POS tagging

I use the result of “main_stan.py” in part-of-speech module for challenge.txt

```
Part-of-speech tagging results:
[[('It', 'PRP'), ('will', 'MD'), ('be', 'VB'), ('in', 'IN'), ('the', 'DT'), ('office', 'NN'), ('fridge', 'NN')]]
```

24) Earley Parser parsing

There is only 1 parse tree for this sentence.



16.Test case 16: limitation example

	Input file and content	Output on console and parse tree diagram	

Input file			Console Output
	data/limitation.txt	Sentences splitting results: ['John intended to eat the apple on the table in his office.'] --- Part-of-speech tagging results: [[('John', 'NNP'), ('intended', 'VBD'), ('to', 'TO'), ('eat', 'VB'), ('the', 'DT'), ('apple', 'NN'), ('on', 'IN'), ('the', 'DT'), ('table', 'NN'), ('in', 'IN'), ('his', 'PRP$'), ('office', 'NN')]] Name entities: {'John'} --- Parsing results: (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (VP[NUM=?n, TENSE='past'] (IV[TENSE='past'] intended) (TO[] to) (VP[NUM=?n, TENSE='inf'] (VP[NUM=?n, TENSE='inf'] (VP[NUM=?n, TENSE='inf'] (TV[TENSE='inf'] eat) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))) (PP[] (IN[] (P[] on)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] table)))))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (PRP[] his) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] office))))))))) (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (VP[NUM=?n, TENSE='past'] (IV[TENSE='past'] intended) (TO[] to) (VP[NUM=?n, TENSE='inf'] (VP[NUM=?n, TENSE='inf'] (TV[TENSE='inf'] eat) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))) (PP[] (IN[] (P[] on)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] table))))))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (PRP[] his) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] office))))))))) (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (VP[NUM=?n, TENSE='past'] (IV[TENSE='past'] intended) (TO[] to) (VP[NUM=?n, TENSE='inf'] (VP[NUM=?n, TENSE='inf'] (TV[TENSE='inf'] eat) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple)))))	

```
      (PP[]
       (IN[] (P[] on))
       (NP[NUM=?n]
        (DT[NUM=?n] (Det[NUM=?n] the))
        (Nom[NUM=?n] (N[NUM='sg'] (NN[] table)))
        (PP[]
         (IN[] (P[] in))
         (NP[NUM=?n]
          (PRP[] his)
          (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[]
office)))))))))))
(S[]
 (NP[NUM='sg']
  (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John))))
 (VP[NUM=?n, TENSE='past']
  (IV[TENSE='past'] intended)
  (TO[] to)
  (VP[NUM=?n, TENSE='inf']
   (TV[TENSE='inf'] eat)
   (NP[NUM=?n]
    (DT[NUM=?n] (Det[NUM=?n] the))
    (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple)))
    (PP[]
     (IN[] (P[] on))
     (NP[NUM=?n]
      (DT[NUM=?n] (Det[NUM=?n] the))
      (Nom[NUM=?n] (N[NUM='sg'] (NN[] table)))
      (PP[]
       (IN[] (P[] in))
       (NP[NUM=?n]
        (PRP[] his)
        (NP[NUM=?n]
         (Nom[NUM=?n] (N[NUM='sg'] (NN[] office))))))))))))
(S[]
 (NP[NUM='sg']
  (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John))))
 (VP[NUM=?n, TENSE='past']
  (VP[NUM=?n, TENSE='past']
   (VP[NUM=?n, TENSE='past']
    (IV[TENSE='past'] intended)
    (TO[] to)
    (VP[NUM=?n, TENSE='inf']
     (TV[TENSE='inf'] eat)
     (NP[NUM=?n]
      (DT[NUM=?n] (Det[NUM=?n] the))
      (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))))
   (PP[]
    (IN[] (P[] on))
    (NP[NUM=?n]
     (DT[NUM=?n] (Det[NUM=?n] the))
     (Nom[NUM=?n] (N[NUM='sg'] (NN[] table))))))
  (PP[]
   (IN[] (P[] in))
   (NP[NUM=?n]
    (PRP[] his)
    (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] office)))))))))
(S[]
 (NP[NUM='sg']
  (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John))))
 (VP[NUM=?n, TENSE='past']
  (VP[NUM=?n, TENSE='past']
   (IV[TENSE='past'] intended)
   (TO[] to)
   (VP[NUM=?n, TENSE='inf']
    (VP[NUM=?n, TENSE='inf']
     (TV[TENSE='inf'] eat)
     (NP[NUM=?n]
      (DT[NUM=?n] (Det[NUM=?n] the))
```

		(Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))))) (PP[] (IN[] (P[] on)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] table))))))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (PRP[] his) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] office)))))))) (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (VP[NUM=?n, TENSE='past'] (VP[NUM=?n, TENSE='past'] (IV[TENSE='past'] intended) (TO[] to) (VP[NUM=?n, TENSE='inf'] (TV[TENSE='inf'] eat) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple))) (PP[] (IN[] (P[] on)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] table)))))))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (PRP[] his) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] office)))))))) (S[] (NP[NUM='sg'] (Nom[NUM='sg'] (NNP[NUM='sg'] (PropN[NUM='sg'] John)))) (VP[NUM=?n, TENSE='past'] (VP[NUM=?n, TENSE='past'] (IV[TENSE='past'] intended) (TO[] to) (VP[NUM=?n, TENSE='inf'] (TV[TENSE='inf'] eat) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] apple)))))) (PP[] (IN[] (P[] on)) (NP[NUM=?n] (DT[NUM=?n] (Det[NUM=?n] the)) (Nom[NUM=?n] (N[NUM='sg'] (NN[] table))) (PP[] (IN[] (P[] in)) (NP[NUM=?n] (PRP[] his) (NP[NUM=?n] (Nom[NUM=?n] (N[NUM='sg'] (NN[] office)))))))))))	
content	John intended to eat the apple on the table in his office.	There are 8 diagrams generated, please see the diagrams below.	Parse Tree Diagram

limitation text result explanation:

25) Sentence splitting

There is 1 sentence in total in limitation.txt, and the result matches.

```
Sentences splitting results:
['John intended to eat the apple on the table in his office.']
```

26) Named Entity Module

I use the union result of “main_stan.py” in named entity module for limitation.txt

```
Name entities:
{'John'}
```

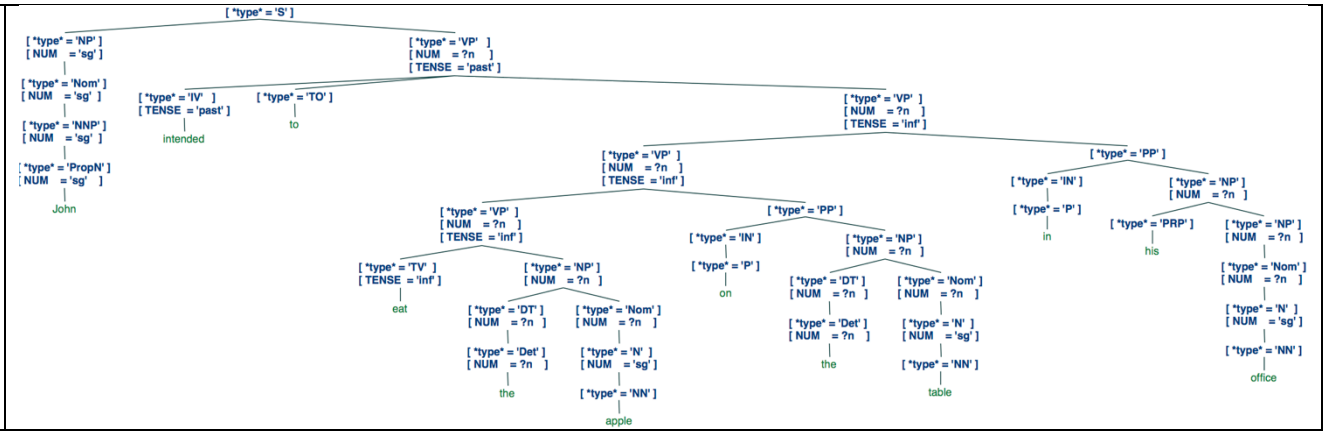
27) POS tagging

I use the result of “main_stan.py” in part-of-speech module for challenge.txt

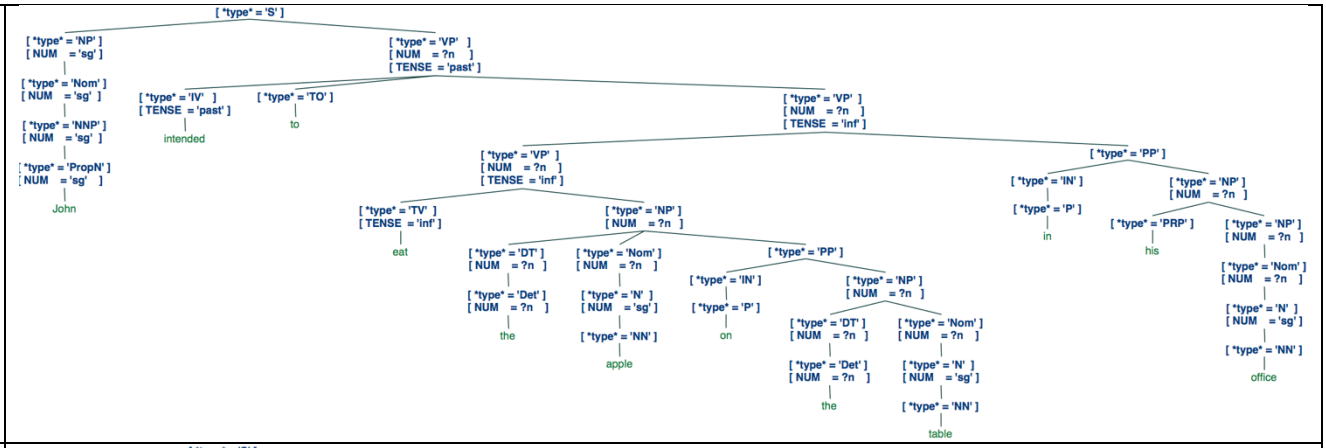
```
Part-of-speech tagging results:
[[('John', 'NNP'), ('intended', 'VBD'), ('to', 'TO'), ('eat', 'VB'), ('the', 'DT'), ('apple', 'NN'), ('on', 'IN'), ('the', 'DT'), ('table', 'NN'), ('in', 'IN'), ('his', 'PRP$'), ('office', 'NN')]]
```

28) Earley Parser parsing

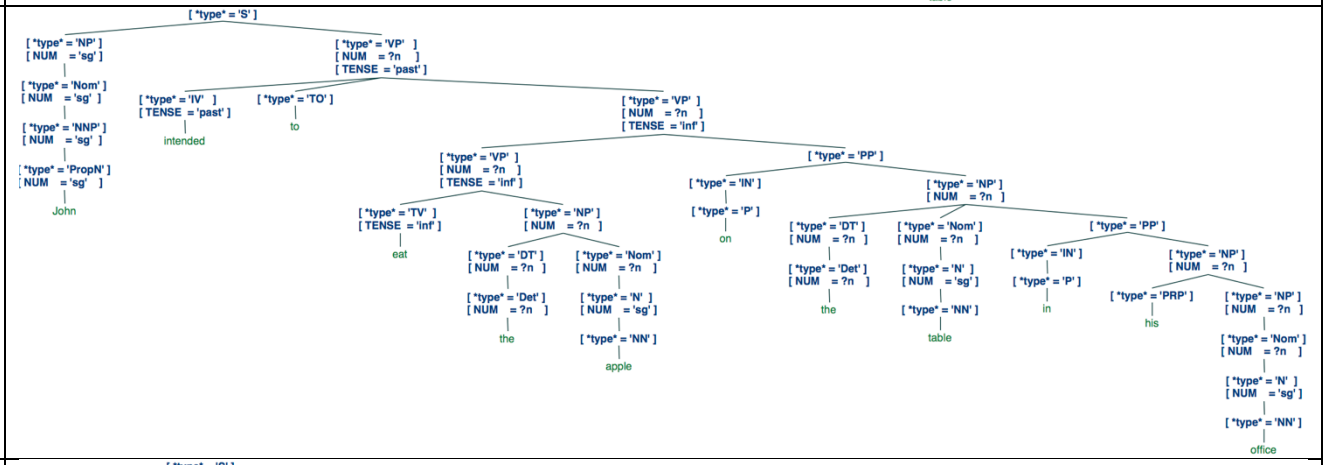
There are 8 parse trees for this sentence, and there are two PP constituent “on the table” and “in his office”. In this example, all 8 parse trees might possible be the acceptable result. Since “on the table” could modify the NOM “apple” or attach to the verb “ate”. And “in his office could modify the NOM “the table” or “ate”. So my grammar generates all available results, and it is possible calculate the probabilities using PCFG in the future and select the parse tree with the largest probability.

Parse tree 1	

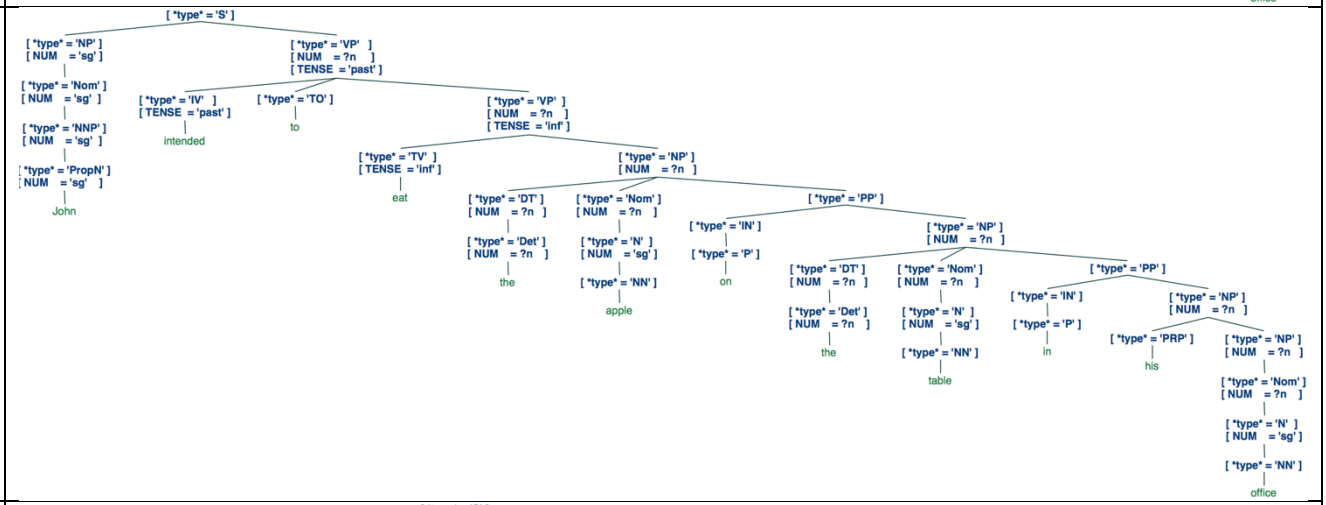
Parse tree 2



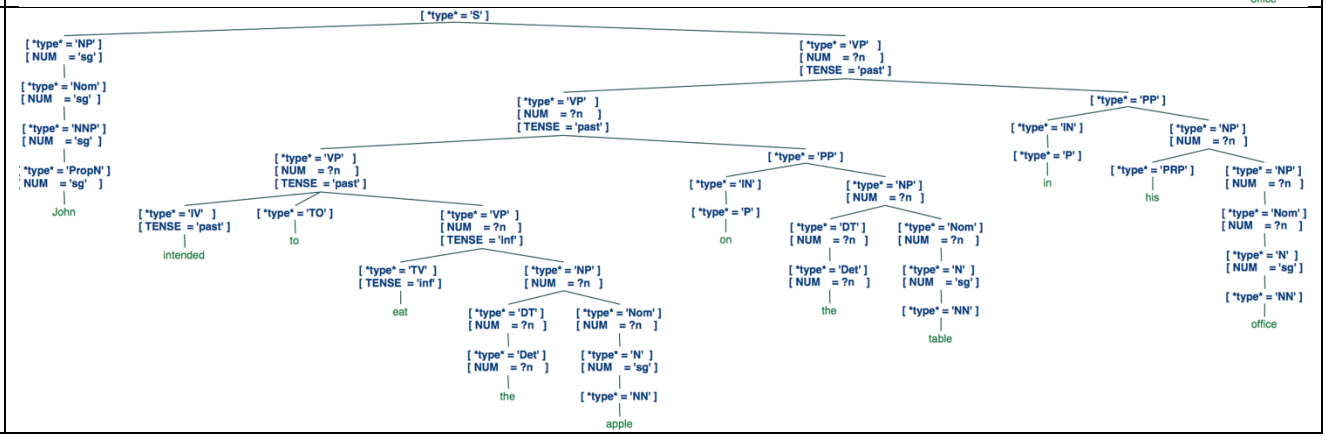
Parse tree 3

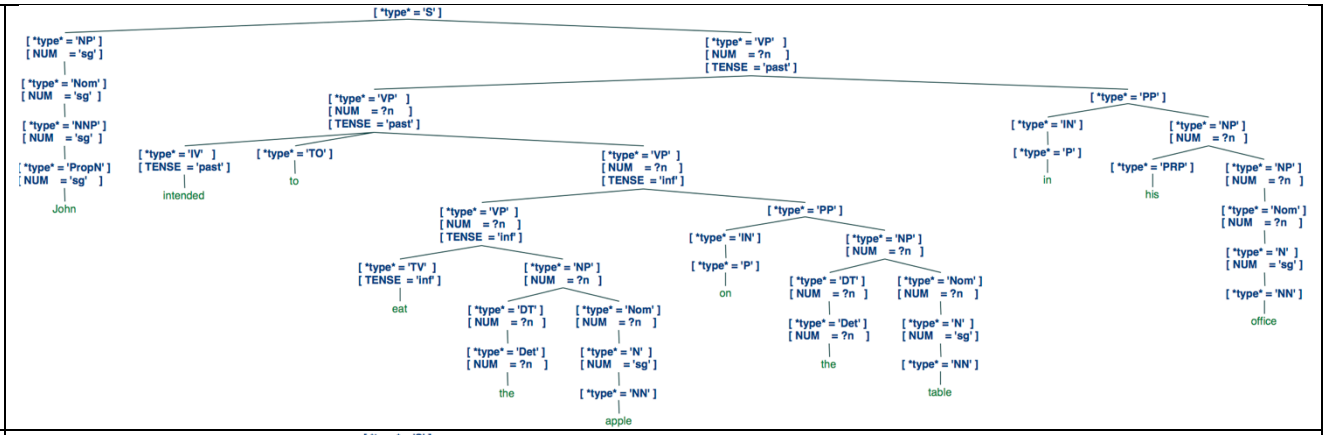
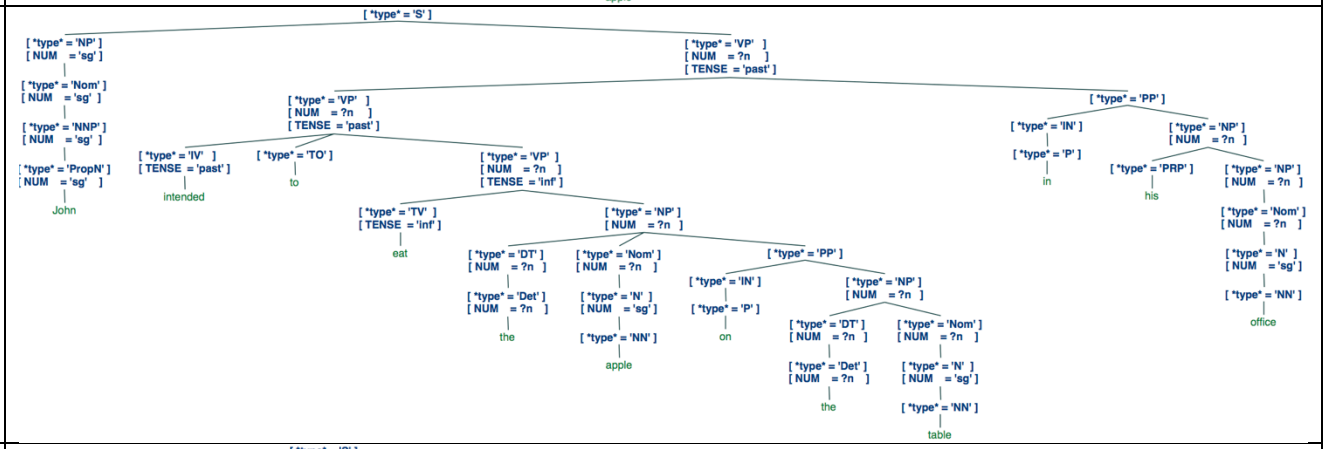


Parse tree 4



Parse tree 5



Parse tree 6	
Parse tree 7	
Parse tree 8