CSE471/598 NL2KR Project

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

I broke the project into 4 phases to Train and Learn. I processed each sentence shown below.

1 Sentence 1

Grammar for Rename data (1)

```
sentence=Rename data
syntaxFile=./examples/sample/syntax.txt
```

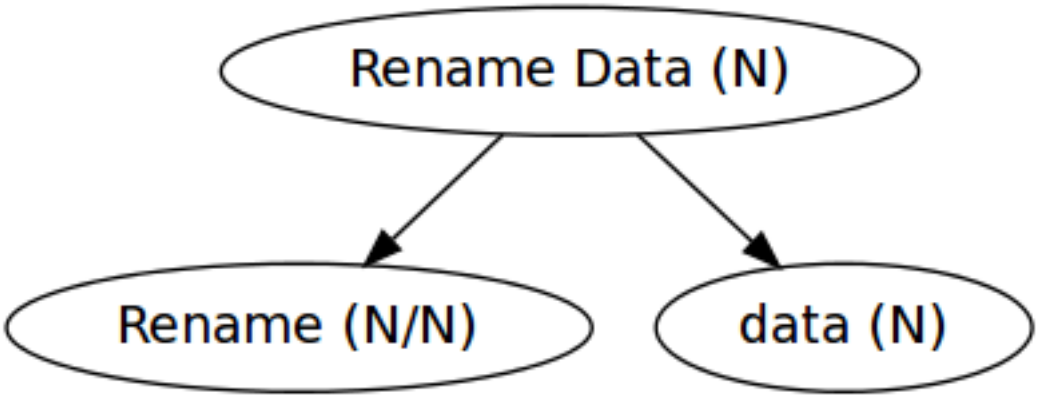
```
Tree number 1:
nl2kr_token("t_1","Rename","(N/N)",-1) #❶
nl2kr_child_left("t_3","t_1")
nl2kr_token("t_3","Rename data","N",-1)
nl2kr_token("t_2","data","N",-1) #❷
nl2kr_child_right("t_3","t_2")
nl2kr_valid_rootNode("t_3")

Tree number 2:
nl2kr_token("t_3","Rename data","((S\\NP)\\((S\\NP)/NP))",-1)
nl2kr_token("t_1","Rename","(N/N)",-1) #❸
nl2kr_child_left("t_4","t_1")
nl2kr_child_left("t_3","t_1")
nl2kr_token("t_4","Rename data","N",-1)
nl2kr_child_right("t_4","t_2")
nl2kr_token("t_2","data","N",-1) #❹
nl2kr_child_right("t_3","t_2")
nl2kr_valid_rootNode("t_3")
nl2kr_valid_rootNode("t_4")

Tree number 3:
nl2kr_token("t_1","Rename","(N/N)",-1) #❺
nl2kr_child_left("t_4","t_1")
nl2kr_child_left("t_3","t_1")
nl2kr_token("t_3","Rename data","(S\\(S/NP))",-1)
nl2kr_token("t_4","Rename data","N",-1)
nl2kr_child_right("t_4","t_2")
nl2kr_token("t_2","data","N",-1) #❻
nl2kr_child_right("t_3","t_2")
nl2kr_valid_rootNode("t_3")
nl2kr_valid_rootNode("t_4")
```

❶ CCG Grammar for Rename agrees across 3 parse trees
❷ CCG Grammar for _data_ agrees across 3 parse trees

The first graph makes logical sense:



Each CCG of the individual words agree so we added this to the syntax file.

```
Rename   (N/N)
data     (N)
```

And add to our training document

```
Rename data      command > rename(data)
```

2 Sentence 2

Next sentence we do the same thing

```
sentence=Print financial audit_report
syntaxFile=./examples/sample/syntax.txt
```

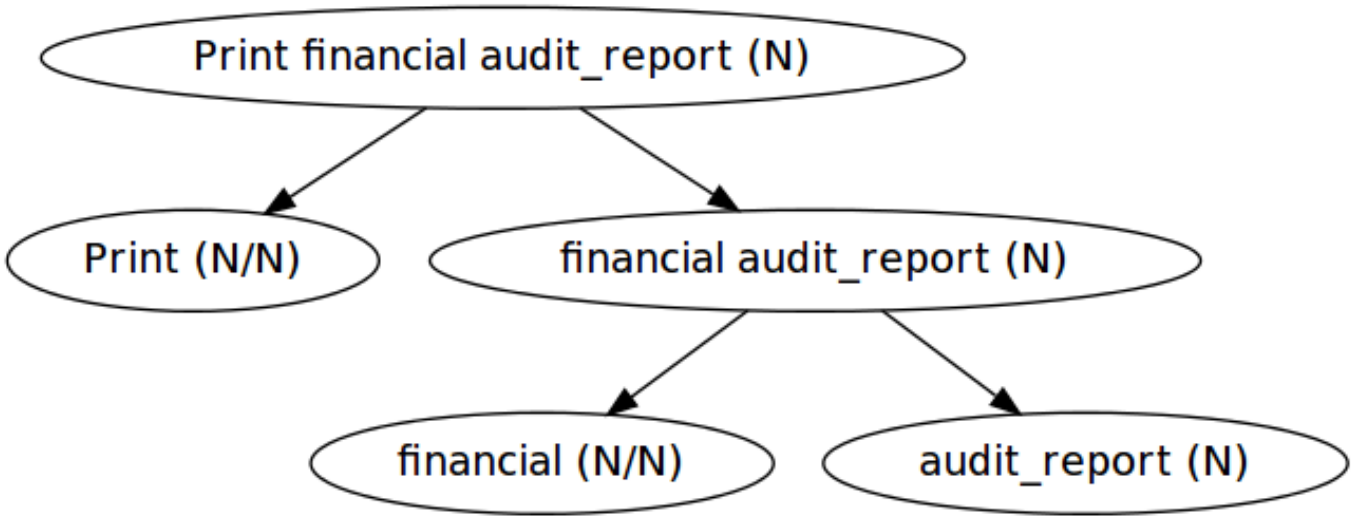
Which generates the following output

```
Tree number 1:
nl2kr_token("t_5","Print financial audit_report","N",-1)
nl2kr_token("t_4","financial audit_report","N",-1)
nl2kr_child_left("t_4","t_1")
nl2kr_token("t_3","Print","(N/N)",-1)
nl2kr_valid_rootNode("t_5")
nl2kr_child_right("t_5","t_4")
nl2kr_child_right("t_4","t_2")
nl2kr_token("t_1","financial","(N/N)",-1)
nl2kr_token("t_2","audit_report","N",-1)
nl2kr_child_left("t_5","t_3")

Tree number 2:
nl2kr_token("t_5","Print financial audit_report","N",-1)
nl2kr_child_right("t_4","t_3")
nl2kr_child_right("t_5","t_1")
nl2kr_child_left("t_5","t_4")
nl2kr_token("t_4","Print financial","(N/N)",-1)
nl2kr_child_left("t_4","t_2")
nl2kr_valid_rootNode("t_5")
nl2kr_token("t_1","audit_report","N",-1)
nl2kr_token("t_3","financial","(N/N)",-1)
nl2kr_token("t_2","Print","((N/N)/(N/N))",-1)

Tree number 3:
nl2kr_token("t_4","financial audit_report","N",-1)
nl2kr_token("t_5","Print financial audit_report","NP",-1)
nl2kr_child_right("t_4","t_1")
nl2kr_token("t_2","Print","(NP/N)",-1)
nl2kr_valid_rootNode("t_5")
nl2kr_child_left("t_4","t_3")
nl2kr_child_right("t_5","t_4")
nl2kr_token("t_1","audit_report","N",-1)
nl2kr_token("t_3","financial","(N/N)",-1)
nl2kr_child_left("t_5","t_2")
```

The first tree makes sense



So we append to our grammar file

```
Print   (N/N)
financial     (N/N)
audit_report  (N)
```

And add to our training document (See phase 1)

```
Print financial audit_report     command > print(audit_report(finance))
```

3 Sentence 3

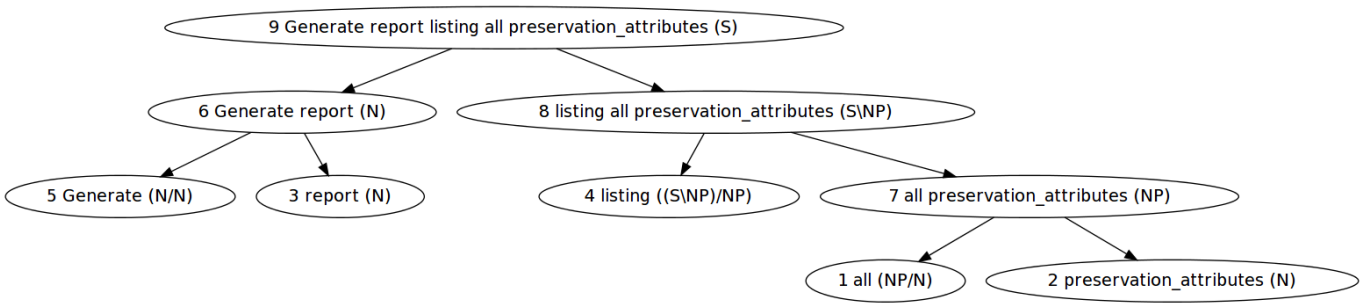
Next sequence we do the same thing

```
sentence=Generate report listing all preservation_attributes
syntaxFile=./examples/sample/syntax.txt
```

Which generates the following output

```
Tree number 1:
nl2kr_token("t_6","Generate report","N",-1)
nl2kr_token("t_9","Generate report listing all preservation_attributes","S",-1)
nl2kr_child_right("t_8","t_7")
nl2kr_token("t_5","Generate","(N/N)",-1)
nl2kr_token("t_2","preservation_attributes","N",-1)
nl2kr_child_left("t_7","t_1")
nl2kr_child_left("t_6","t_5")
nl2kr_child_left("t_8","t_4")
nl2kr_token("t_1","all","(NP/N)",-1)
nl2kr_child_right("t_7","t_2")
nl2kr_child_right("t_9","t_8")
nl2kr_token("t_7","all preservation_attributes","NP",-1)
nl2kr_token("t_3","report","N",-1)
nl2kr_token("t_4","listing","((S\\NP)/NP)",-1)
nl2kr_child_left("t_9","t_6")
nl2kr_child_right("t_6","t_3")
nl2kr_token("t_8","listing all preservation_attributes","(S\\NP)",-1)
nl2kr_valid_rootNode("t_9")
```

The tree makes sense



So we append to our syntax file

```
Generate (N/N)
report (N)
listing ((S\\NP)/NP)
all (NP/N)
preservation_attributes (N)
```

And add to our training document

```
Generate report listing all preservation_attributes     command > generate(report(list( ↩
    preservation_attributes)))
```

4 Sentence 4

Next sentence

```
sentence=Transfer data to new storage
syntaxFile=./examples/sample/syntax.txt
```

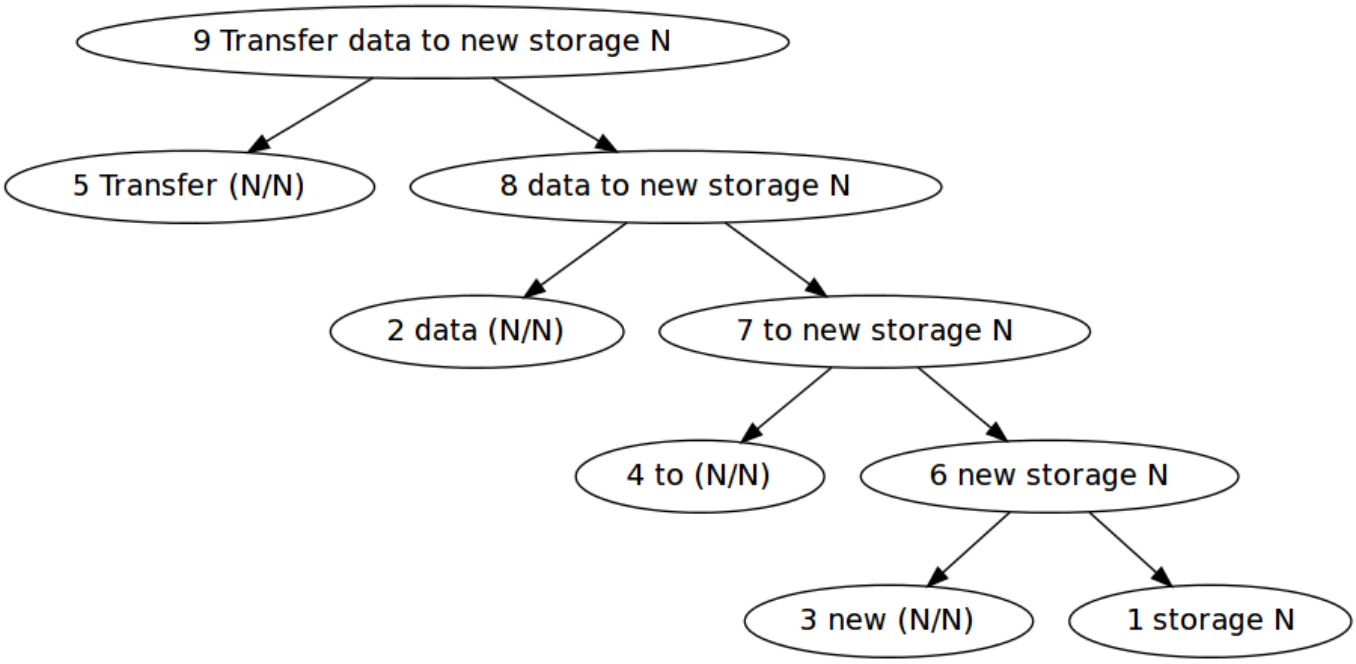
Which generates the following output

```
Tree number 1:
nl2kr_token("t_5","Transfer","(N/N)",-1)
nl2kr_child_right("t_8","t_7")
nl2kr_child_left("t_9","t_5")
nl2kr_token("t_1","storage","N",-1)
nl2kr_token("t_8","data to new storage","N",-1)
nl2kr_token("t_3","new","(N/N)",-1)
nl2kr_child_left("t_6","t_3")
nl2kr_child_left("t_7","t_4")
nl2kr_token("t_2","data","(N/N)",-1)
nl2kr_token("t_6","new storage","N",-1)
nl2kr_child_right("t_6","t_1")
nl2kr_child_right("t_9","t_8")
nl2kr_token("t_7","to new storage","N",-1)
nl2kr_token("t_4","to","(N/N)",-1)
nl2kr_child_right("t_7","t_6")
nl2kr_child_left("t_8","t_2")
nl2kr_token("t_9","Transfer data to new storage","N",-1)
nl2kr_valid_rootNode("t_9")

Tree number 2:
nl2kr_token("t_5","Transfer","(N/N)",-1)
nl2kr_token("t_3","data","N",-1)
nl2kr_token("t_8","to new storage","(NP\\NP)",-1)
nl2kr_child_left("t_8","t_1")
nl2kr_child_left("t_6","t_5")
```

```
nl2kr_token("t_7","new storage","N",-1)
nl2kr_token("t_2","storage","N",-1)
nl2kr_token("t_9","Transfer data to new storage","NP",-1)
nl2kr_child_right("t_7","t_2")
nl2kr_child_left("t_7","t_4")
nl2kr_token("t_6","Transfer data","N",-1)
nl2kr_token("t_1","to","((NP\\NP)/NP)",-1)
nl2kr_child_right("t_8","t_7")
nl2kr_child_left("t_9","t_6")
nl2kr_token("t_4","new","(N/N)",-1)
nl2kr_child_right("t_6","t_3")
nl2kr_child_right("t_9","t_8")
nl2kr_valid_rootNode("t_9")
```

The tree makes sense



So we append to our syntax file

```
Transfer (N/N)
data (N/N)
to (N/N)
new (N/N)
storage (N)
```

And add to our training document

```
Transfer data to new storage    command > transfer(data, storage(new))
```

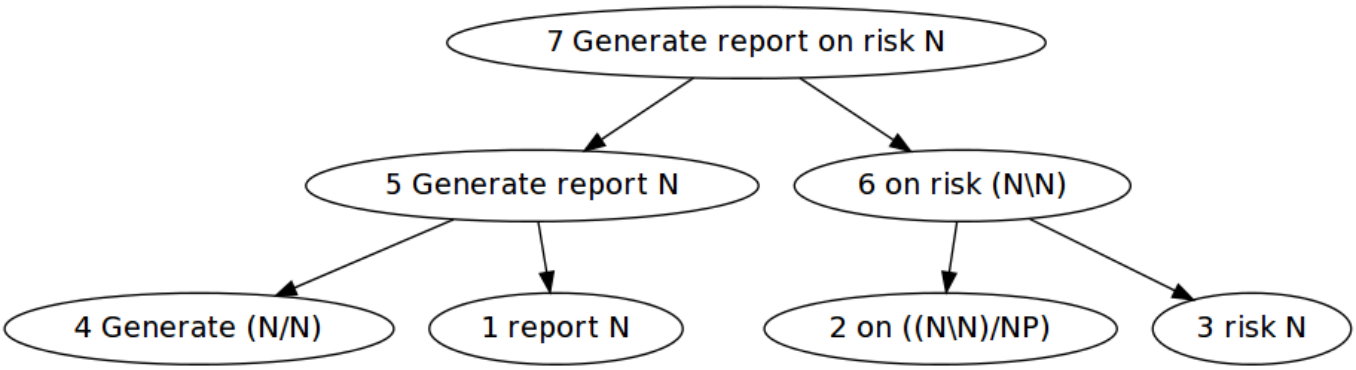
5 Sentence 5

```
sentence=Generate report on risk
syntaxFile=./examples/sample/syntax.txt
```

Which generates the following output

```
l2kr_child_right("t_5","t_1")
nl2kr_child_left("t_5","t_4")
nl2kr_token("t_6","on risk","(N\\N)",-1)
nl2kr_token("t_3","risk","N",-1)
nl2kr_token("t_7","Generate report on risk","N",-1)
nl2kr_token("t_4","Generate","(N/N)",-1)
nl2kr_token("t_2","on","((N\\N)/NP)",-1)
nl2kr_token("t_1","report","N",-1)
nl2kr_child_left("t_7","t_5")
nl2kr_valid_rootNode("t_7")
nl2kr_child_right("t_7","t_6")
nl2kr_child_left("t_6","t_2")
nl2kr_child_right("t_6","t_3")
nl2kr_token("t_5","Generate report","N",-1)
```

The tree makes sense



So we append to our grammar gile

```
Generate (N/N)
report N
on ((N\\N)/NP)
risk N
```

And add to our training file

```
Generate report on risk command > generate(report(risk))
```

6 Building Lambda Definitions

Using the defined command parsing, we can infer the function application

```
print(audit_report(finance))
```

We can reverse the lambda function application to get the root expressions

```
print(audit_report(finance))
print(\f.audit_report(f)@finance)
\a.print(a)@\f.(audit_report(f)@finance
```

Thus the individual expressions are

```
#x.print(x)
#x.audit_report(x)
finance
```

Which we can add to our dictionary file.

```
print (N/N) #x.print(x)
financial (N/N) financial
audit_report (N) #x.audit_report(x)
```

7 Learning

Now that we have our 5 sentences, and we trained our system sufficiently, we can choose 3 sentences for learning. I choose the following sentences:

Chosen sentences

```
Rename collection
Print staff_experience report
Transfer ownership to rods
```

Learning Configuration

```
Ldata=../PartA/wright_train.txt
Ldictionary=../PartA/wright_dict.txt
Lsyntax=../PartA/wright_syntax.txt
Loutput=../PartA/wright_train.out
```

After correcting a few syntax errors in the dictionary file, generates the following lexicon file

```
financial       [N/N]   financial       0.01
to      [N/N]   to      0.01
data    [N/N]   data    0.01
data    [N]     data    0.01
audit_report    [N]     #x.audit_report(x)      0.01
audit_report    [N]     audit_report    0.01
Print   [N/N]   #x.print(x)     0.01
Transfer        [N/N]   #x.#y.transfer(x,y)     0.01
risk    [N]     risk    0.01
Rename  [N/N]   #x.rename(x)    0.01
new     [N/N]   new     0.01
listing [(S\NP)/NP]     #x.list(x)      0.01
report  [N]     #x.report(x)    0.01
report  [N]     report  0.01
Generate        [N/N]   #x.generate(x)  0.01
storage [N]     storage 0.01
preservation_attributes [N]     preservation_attributes 0.01
```

To which we can test new files. This resulted in the following error:

```
TestingProcess: 3 testing data read.
******Parsing Sentences ...
Parsing test sentence: Rename collection
Expected Representation: command > rename(collection)
Predicted Result: command > rename(collection)
Correct Prediction

Parsing test sentence: Print staff_experience report
Expected Representation: command > print(report(staff_experience))
Predicted Result: command > print(report(staff_experience))
Correct Prediction

Parsing test sentence: Transfer ownership to rods
Expected Representation: command > transfer(ownership, rods)
```

```
java.io.FileNotFoundException: resources/aspccgtk-parser/output/ ↩
    output_Transfer_ownership_to_rods.asp (No such file or directory)
        at java.io.FileInputStream.open(Native Method)
        at java.io.FileInputStream.<init>(FileInputStream.java:138)
        at java.io.FileInputStream.<init>(FileInputStream.java:97)
        at java.io.FileReader.<init>(FileReader.java:58)
        at bioai.ccgprocessor.ccgparsing.aspccgtkParser.ASPCCGTKWrapper.callASPCCGParser( ↩
            ASPCCGTKWrapper.java:185)
        at bioai.ccgprocessor.ccgparsing.aspccgtkParser.ASPCCGParser.parse(ASPCCGParser. ↩
            java:32)
        at bioai.ccgprocessor.pccg.parser.PCCGParser.parseTable(PCCGParser.java:172)
        at bioai.ccgprocessor.pccg.parser.PCCGParser.parse(PCCGParser.java:251)
        at bioai.ccgprocessor.translation.TranslationProcess.testPair(TranslationProcess. ↩
            java:196)
        at bioai.ccgprocessor.translation.TranslationProcess.test(TranslationProcess.java ↩
            :112)
        at bioai.ccgprocessor.tests.scripts.NL2KR_TTest.main(NL2KR_TTest.java:90)
java.lang.NullPointerException
        at bioai.ccgprocessor.ccgparsing.aspccgtkParser.ASPCCGParser.parse(ASPCCGParser. ↩
            java:37)
        at bioai.ccgprocessor.pccg.parser.PCCGParser.parseTable(PCCGParser.java:172)
        at bioai.ccgprocessor.pccg.parser.PCCGParser.parse(PCCGParser.java:251)
        at bioai.ccgprocessor.translation.TranslationProcess.testPair(TranslationProcess. ↩
            java:196)
        at bioai.ccgprocessor.translation.TranslationProcess.test(TranslationProcess.java ↩
            :112)
        at bioai.ccgprocessor.tests.scripts.NL2KR_TTest.main(NL2KR_TTest.java:90)
java.lang.NullPointerException
        at bioai.ccgprocessor.pccg.parser.PCCGParser.parse(PCCGParser.java:252)
        at bioai.ccgprocessor.translation.TranslationProcess.testPair(TranslationProcess. ↩
            java:196)
        at bioai.ccgprocessor.translation.TranslationProcess.test(TranslationProcess.java ↩
            :112)
        at bioai.ccgprocessor.tests.scripts.NL2KR_TTest.main(NL2KR_TTest.java:90)
```

Thus we need to provide more context to the training engine.

Create a new learning file

```
Ldata=../PartA/wright3_learn.txt
Ldictionary=../PartA/wright3_dict.txt
Lsyntax=../PartA/wright3_syntax.txt
Loutput=../PartA/wright3_policy.out
```

We want to teach the system what transfer means so we provide the dictionary for some nouns

```
ownership       N       ownership
rods    N       ownership
```

Run the learning engine again to learn the new vocabulary.

We repaired the error, but still no parse tree.

```
student@student-VirtualBox ~/Desktop/workspaces/471_project/NL2KR-System $ ./NL2KR-T.sh  ↩
    wright_train.conf
******Reading test data: ../PartA/wright_train.txt
Test line: 0
Test line: 1
Test line: 2
******Reading lexicon file: ../PartA/wright_learn.out
Line[0] Syntax:[[N/N]] Semantics:[financial]
Line[1] Syntax:[[N/N]] Semantics:[to]
Line[2] Syntax:[[N/N]] Semantics:[data]
```

```
Line[3] Syntax:[[N]] Semantics:[data]
Line[4] Syntax:[[N]] Semantics:[#x.audit_report(x)]
Line[5] Syntax:[[N]] Semantics:[audit_report]
Line[6] Syntax:[[N/N]] Semantics:[#x.print(x)]
Line[7] Syntax:[[N/N]] Semantics:[#x.#y.transfer(x,y)]
Line[8] Syntax:[[N]] Semantics:[risk]
Line[9] Syntax:[[N/N]] Semantics:[#x.rename(x)]
Line[10] Syntax:[[N/N]] Semantics:[new]
Line[11] Syntax:[[(S\NP)/NP]] Semantics:[#x.list(x)]
Line[12] Syntax:[[N]] Semantics:[#x.report(x)]
Line[13] Syntax:[[N]] Semantics:[report]
Line[14] Syntax:[[N/N]] Semantics:[#x.generate(x)]
Line[15] Syntax:[[N]] Semantics:[storage]
Line[16] Syntax:[[N]] Semantics:[preservation_attributes]
TestingProcess: 3 testing data read.
******Parsing Sentences ...
Parsing test sentence: Rename collection
Expected Representation: command > rename(collection)
Predicted Result: command > rename(collection)
Correct Prediction

Parsing test sentence: Print staff_experience report
Expected Representation: command > print(report(staff_experience))
Predicted Result: command > print(report(staff_experience))
Correct Prediction

Parsing test sentence: Transfer ownership to rods
Expected Representation: command > transfer(ownership, rods)
Generalizing ownership = null
Generalizing ownership = null
Correct output does not exist in the parse result
Predicted Result:null
Wrong Prediction

Correct Predictions : 2/3
Incorrect Predictions : 0/3
Predictions for sentences having unknown Expected Representations : 0/0
No Predictions : 1/3
Total evaluation costs: 00h:00m:47s:012ms
```

We have the policy learnt from the previous step, we can append this to the overall lexicon. The resulting policy file is:

```
Refinancial     [N/N]   financial       0.01
to      [N/N]   to      0.01
data    [N/N]   data    0.01
data    [N]     data    0.01
audit_report    [N]     #x.audit_report(x)      0.01
audit_report    [N]     audit_report    0.01
Print   [N/N]   #x.print(x)     0.01
Transfer        [N/N]   #x.#y.transfer(x,y)     0.01
risk    [N]     risk    0.01
Rename  [N/N]   #x.rename(x)    0.01
new     [N/N]   new     0.01
listing [(S\NP)/NP]     #x.list(x)      0.01
report  [N]     #x.report(x)    0.01
report  [N]     report  0.01
Generate        [N/N]   #x.generate(x)  0.01
storage [N]     storage 0.01
preservation_attributes [N]     preservation_attributes 0.01
ownership       [N]     ownership       0.01
rods    [N]     rods    0.01
rods    [N]     ownership       0.01
```

Still unable to parse the result

```
******Reading test data: ../PartA/wright_train.txt
Test line: 0
Test line: 1
Test line: 2
******Reading lexicon file: ../PartA/wright_learn.out
Line[0] Syntax:[[N/N]] Semantics:[financial]
Line[1] Syntax:[[N/N]] Semantics:[to]
Line[2] Syntax:[[N/N]] Semantics:[data]
Line[3] Syntax:[[N]] Semantics:[data]
Line[4] Syntax:[[N]] Semantics:[#x.audit_report(x)]
Line[5] Syntax:[[N]] Semantics:[audit_report]
Line[6] Syntax:[[N/N]] Semantics:[#x.print(x)]
Line[7] Syntax:[[N/N]] Semantics:[#x.#y.transfer(x,y)]
Line[8] Syntax:[[N]] Semantics:[risk]
Line[9] Syntax:[[N/N]] Semantics:[#x.rename(x)]
Line[10] Syntax:[[N/N]] Semantics:[new]
Line[11] Syntax:[[(S\NP)/NP]] Semantics:[#x.list(x)]
Line[12] Syntax:[[N]] Semantics:[#x.report(x)]
Line[13] Syntax:[[N]] Semantics:[report]
Line[14] Syntax:[[N/N]] Semantics:[#x.generate(x)]
Line[15] Syntax:[[N]] Semantics:[storage]
Line[16] Syntax:[[N]] Semantics:[preservation_attributes]
Line[17] Syntax:[[N]] Semantics:[ownership]
Line[18] Syntax:[[N]] Semantics:[rods]
Line[19] Syntax:[[N]] Semantics:[ownership]
TestingProcess: 3 testing data read.
******Parsing Sentences ...
Parsing test sentence: Rename collection
Expected Representation: command > rename(collection)
Predicted Result: command > rename(collection)
Correct Prediction

Parsing test sentence: Print staff_experience report
Expected Representation: command > print(report(staff_experience))
Predicted Result: command > print(report(staff_experience))
Correct Prediction

Parsing test sentence: Transfer ownership to rods
Expected Representation: command > transfer(ownership, rods)
Correct output does not exist in the parse result
Predicted Result:null
Wrong Prediction

Correct Predictions : 2/3
Incorrect Predictions : 0/3
Predictions for sentences having unknown Expected Representations : 0/0
No Predictions : 1/3
Total evaluation costs: 00h:00m:47s:729ms
```

Trying a new third sentence. Trained the result with file phase 4.

```
Rename collection
Print staff_experience report
file is master_copy ❶
```

❶ Changed sentence.

Still cannot learn the third sentence

```
******Reading test data: ../PartA/wright_train.txt
```

```
Test line: 0
Test line: 1
Test line: 2
******Reading lexicon file: ../PartA/wright_learn.out
Line[0] Syntax:[[N/N]] Semantics:[financial]
Line[1] Syntax:[[N/N]] Semantics:[to]
Line[2] Syntax:[[N/N]] Semantics:[data]
Line[3] Syntax:[[N]] Semantics:[data]
Line[4] Syntax:[[N]] Semantics:[#x.audit_report(x)]
Line[5] Syntax:[[N]] Semantics:[audit_report]
Line[6] Syntax:[[N/N]] Semantics:[#x.print(x)]
Line[7] Syntax:[[N/N]] Semantics:[#x.#y.transfer(x,y)]
Line[8] Syntax:[[N]] Semantics:[risk]
Line[9] Syntax:[[N/N]] Semantics:[#x.rename(x)]
Line[10] Syntax:[[N/N]] Semantics:[new]
Line[11] Syntax:[[(S\NP)/NP]] Semantics:[#x.list(x)]
Line[12] Syntax:[[N]] Semantics:[#x.report(x)]
Line[13] Syntax:[[N]] Semantics:[report]
Line[14] Syntax:[[N/N]] Semantics:[#x.generate(x)]
Line[15] Syntax:[[N]] Semantics:[storage]
Line[16] Syntax:[[N]] Semantics:[preservation_attributes]
Line[17] Syntax:[[N]] Semantics:[ownership]
Line[18] Syntax:[[N]] Semantics:[rods]
Line[19] Syntax:[[N]] Semantics:[ownership]
Line[20] Syntax:[[N]] Semantics:[master_copy]
Line[21] Syntax:[[N]] Semantics:[file]
TestingProcess: 3 testing data read.
******Parsing Sentences ...
Parsing test sentence: Rename collection
Expected Representation: command > rename(collection)
Predicted Result: command > rename(collection)
Correct Prediction

Parsing test sentence: Print staff_experience report
Expected Representation: command > print(report(staff_experience))
Predicted Result: command > print(report(staff_experience))
Correct Prediction

Parsing test sentence: file is master_copy
Expected Representation: master_copy(file)
Generalizing is = null
Correct output does not exist in the parse result
Predicted Result:null
Wrong Prediction

Correct Predictions : 2/3
Incorrect Predictions : 0/3
Predictions for sentences having unknown Expected Representations : 0/0
No Predictions : 1/3
Total evaluation costs: 00h:00m:47s:085ms
```

At this point, I cannot determine what I am missing in the third sentence.

8 Conclusion

In this project, worked out some manual CCG grammars to verify that the tree's generated by the tool made sense. Several times the CCG tree I generated was incorrect, and the multiple parse trees given by the tool demonstrated my error. Therefore the biggest return I got from this project was the CCG grammar parsing. I faced numerous errors with the other tools, and it was difficult to work through a stack of java exceptions to figure out the underlying problem. Otherwise, it was an interesting project.