Natural Language Processing Homework 4

Katie Chang, Willis Wang

March 25, 2016

README

1 State of the Art Parser

Discussion

It is interesting to see how the different parsers from this question and the extra credit question deal with semantically ambiguous sentences. Let us take the sentence that we have all grown fond of:

> That the president ate the pickle perplexed Sally.

In Figure 1, we can see that this is wrong because "That" cannot be a NP. Ambiguity in a sentence's semantics will often stump a parser and lead to an incorrect parse. The Stanford parser gives us the same parse tree.

We also tested with the sentence

> The plans to raise income tax the imagination.

Here, the two parsers came up with a slightly different parse tree. Stanford's parser (Figure 2) correctly took "The plans" to be a noun phrase, whereas the Berkeley parser (Figure 3) took the word "plans" to be a verb phrase.

1.1 Extra Credit

TurboParser does not give us a tree that is rooted at a root node. Instead, the tree separates phrases and seems to use the end words of each phrase to

```
(ROOT
  (S
    (NP
      (NP (DT That))
      (NP (DT the) (NN president)))
    (VP (VBD ate)
      (SBAR
        (S
          (NP (DT the) (NN pickle))
          (VP (VBD perplexed)
            (NP (NNP Sally))))))
    (. .)))
```

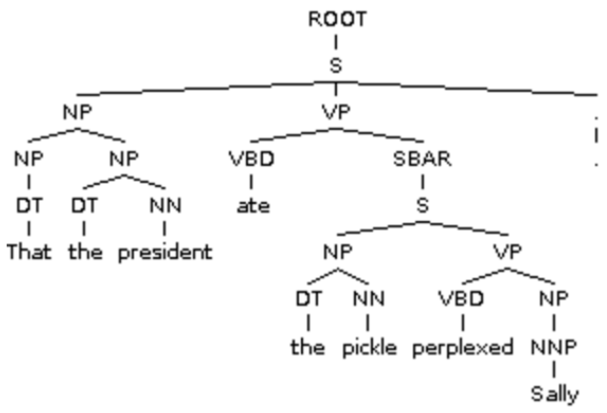
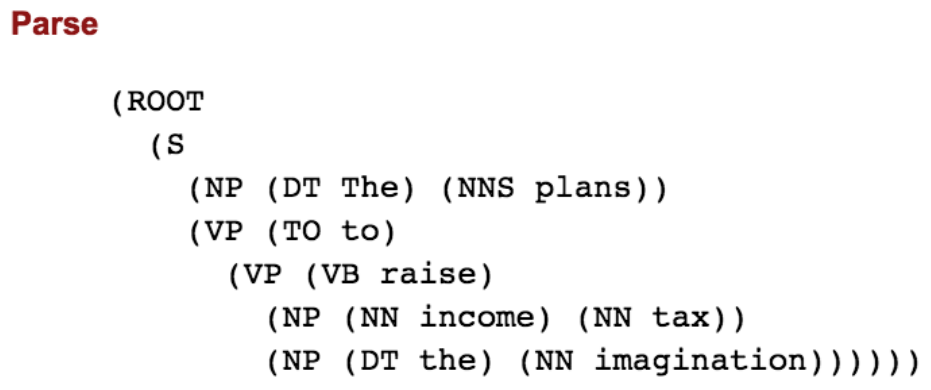


Figure 1: Berkeley Parser on a sentence.


Parse

```
(ROOT
  (S
    (NP (DT The) (NNS plans))
    (VP (TO to)
      (VP (VB raise)
        (NP (NN income) (NN tax))
        (NP (DT the) (NN imagination))))))
```

Figure 2: Stanford Parser on a second sentence.

connect each phrase. In Figure 4, there is no tree root, and the nodes VB, NNS, and VBP are in between each distinct phrase.

The Link Grammar Parser does something interesting, as shown in Figure 5: it parses the sentence and identifies links between phrases / words in the sentence. Using this parse, the software creates a constituent tree, which looks much like the trees that we produced in homework 1.

2 Earley Parser Part 1

Run:

```
python GrammarParsers.py papa.gr papa.sen
```

Our output is like so:

```
Processing: Papa ate the caviar with a spoon
(ROOT (S (NP Papa)
        (VP (VP (V ate)
                (NP (Det the)
                    (N caviar)))
            (PP (P with)
                (NP (Det a)
                    (N spoon))))))
10.2173230517 Time Elapsed: 0:00:00.002555
```

```
(ROOT
  (S
    (NP (DT The))
    (VP (VBZ plans)
      (S
        (VP (TO to)
          (VP (VB raise)
            (NP (NN income) (NN tax))
            (NP (DT the) (NN imagination))))))))
```

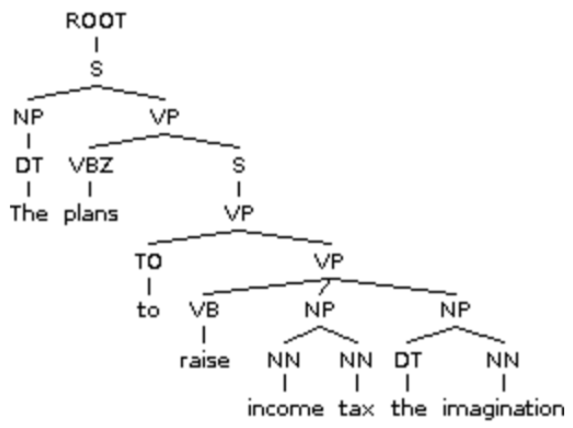


Figure 3: Berkeley Parser on a second sentence.

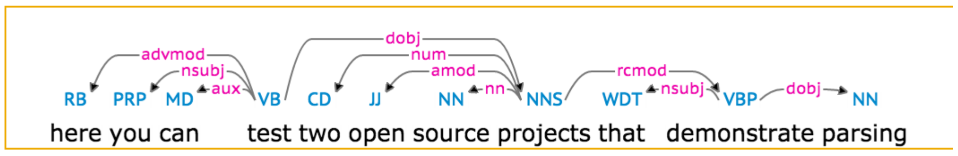
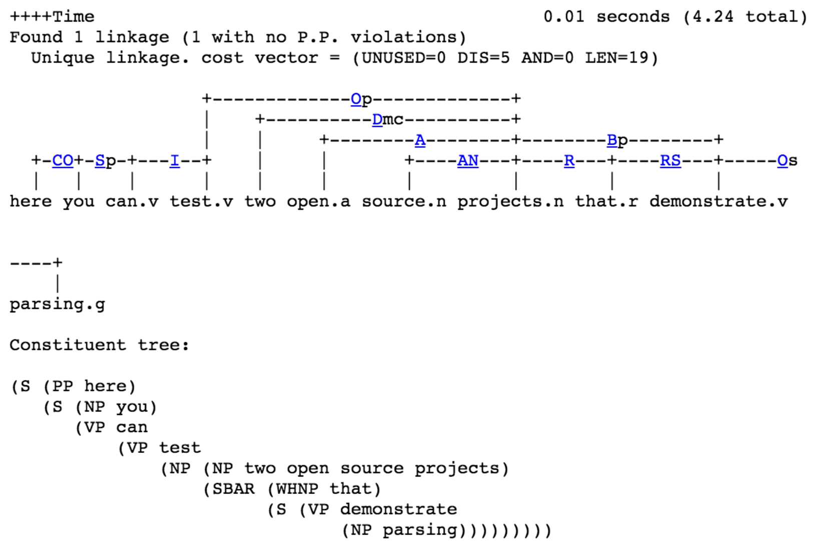

dobj
advmod
num
nsubj
amod
aux
nn
rcmod
nsubj
dobj
RB
PRP
MD
VB
CD
JJ
NN
NNS
WDT
VBP
NN
here you can test two open source projects that demonstrate parsing

Figure 4: TurboParser Tree Diagram of a sample sentence.



```
++++Time                                          0.01 seconds (4.24 total)
Found 1 linkage (1 with no P.P. violations)
  Unique linkage. cost vector = (UNUSED=0 DIS=5 AND=0 LEN=19)

                  +-------------Op-------------+
                  |    +----------Dmc----------+
                  |    |     +--------A--------+--------Bp--------+
  +-CO+-Sp-+---I--+    |     |       +----AN---+----R---+----RS---+-----Os
  |   |    |      |    |     |       |         |        |         |
here you can.v test.v two open.a source.n projects.n that.r demonstrate.v


----+
    |
parsing.g

Constituent tree:

(S (PP here)
   (S (NP you)
      (VP can
          (VP test
              (NP (NP two open source projects)
                  (SBAR (WHNP that)
                        (S (VP demonstrate
                               (NP parsing))))))))
```

Figure 5: LinkGrammarParser Tree Diagram of a sample sentence.

Duplicate check in O(1): We are checking within the relevant hashed column to see if it has the key we want to enqueue yet. If it already exists in our chart, then we do not enqueue this rule to our chart.

Adding entry in O(1): We have an enqueue method inside our Chart class. Here, we keep track of what column we are in currently. When we initialize our Chart, we keep a list of columns. It appends the current rule we want to add, to the column list with an append method call.

We are constantly checking for weight updates in our parseSentence method, when attaching constituents with customers.

3 Earley Parser 2.0

Speedups used:

1. Using Pypy 2. Keeping track of categories already predicted

Processing: Papa ate the caviar with a spoon

```
(ROOT (S (NP Papa)
         (VP (VP (V ate)
                 (NP (Det the)
                    (N caviar)))
            (PP (P with)
                (NP (Det a)
                    (N spoon))))))
```

10.2173230517 Time Elapsed: 0:00:00.002718

wallstreet.sen took ????many seconds to run with this sped-up Earley Parser.

Used overleaf.com to generate LaTeX document.