Advanced Methods in NLP Lecturer: Jonathan Berant

Home Assignment 4: Syntactic Parsing

Due Date: *May 28, 2019*

This assignment is heavily adapted from Yoav Goldberg who adapted it from Jason Eisner.

In this home assignment you will write grammars and implement a toy syntactic parser. You are porivded with a basic grammar and supporting code.

Your code should run properly on Python 2.7 on linux (it doesn't have to run on Python 3 and above). It must run on `nova.cs.tau.ac.il` using `/usr/bin/python`.

Create a zip file named `<id1>_<id2>.zip` (where id1 refers to the ID of the first student) and submit your solution on Moodle. Each group should submit the zip file only once. The zip should include a text file with an e-mail of one of the students.

1 Basic grammar

We provide you with python source files `PCFG.py` and `generate.py`, and a basic grammar file `grammar.txt`. Each line in the grammar file describes a grammar rule:

- The rule's weight.
- The rule's left hand side - a non-terminal symbol.
- The rule's right hand side - a sequence of one or more non-terminal and terminal symbols.

You can generate sentences from the grammar using the `generate.py` program: `python generate.py grammar.txt`. Note that you can generate multiple sentences at once by running: `python generate.py grammar.txt -n <#generated sentences>`. Make sure you understand the code and the basic grammar.

(a) Why does the program generate so many long sentences? Specifically, what grammar rule is responsible and why? What is special about this rule?

(b) The grammar allows multiple adjectives, as in *"the fine perplexed pickle"*. Why do the generated sentences do this so rarely?

(c) The grammar format allows specifying different weights to different rules. Which numbers should you modify to fix the problems in (a) and (b), making the sentences shorter and the adjectives more frequent? Verify your answer by generating from the grammar. Discuss your solution (which rules and why). Submit your new grammar as `grammar1.txt`.

(d) Implement the function `gentree` in `PCFG.py`. The function receives a symbol and generates a derivation tree from that symbol.

Once you implemented this function, given the switch `-t` in commandline, the program `generate.py` should not generate sentences, but the tree structures that generated the sentences. For example, when invoked as: `python generate.py grammar.txt -n 1 -t` instead of just printing *"the floor kissed the delicious chief of staff"* it should print the more elaborate version:

```
(ROOT (S (NP (Det the) (Noun floor)) (VP (Verb kissed) (NP (Det the) (Noun (Adj delicious) (Noun chief of staff))))) .)
```

The bracketed expression can be visualized in tree form using web-based visualizers like `http://christos-c.com/treeviewer/`. This sort of output can be useful when debugging your grammar – understanding which rules are responsible for what structures.

Hint: You don't have to represent a tree as an object in memory, only print the parentheses and nonterminals correctly.

2 CKY parser

Implement the CKY algorithm (CKY slides, slide 30). The algorithm receives a sentence to parse and a grammar, and returns the derivation of the sentence with the highest probability. The derivation returned by the function should be in the format described in question (d) in the section above. In case a sentence cannot be parsed by the grammar, the function should return the string "FAILED TO PARSE!". You should implement 2 versions of CKY algorithm:

(a) The parser should assume that the input grammar is in Chomsky Normal Form (CNF). Fill your implementation in the function `cnf_cky` in `cky.py`.

(b) Extend your implementation to handle rules of the form: $X \rightarrow \sigma_1\sigma_2...\sigma_n$, where $\sigma_i \in \Sigma$ and $n \geq 1$. Fill your implementation in the function `non_cnf_cky` in `cky.py`

3 Extending the Grammar

In this section you will extend your grammar to support more linguistic phenomena.

While developing your grammar you can debug it with your CKY parser: run the parser with your grammar on a set of sentences. The parser should parse the correct sentences, and fail to parse incorrect ones. You can choose on which version of the parser to test your grammar, but in any case make sure your submission includes your grammars in CNF form.

Advice 1: It is easier to write non-CNF grammars, and then convert them to CNF (See syntactic parsing slides, slides 20-22).

Advice 2: If you want to see the effect of grammar rules, you can upweight their probability so that they trigger more often.

(a) Extend your grammar so it can also generate the types of phenomena illustrated in the following sentences (it must still parse all of the sentences it could parse before):

(a) Sally ate a sandwich .

(b) Sally and the president wanted and ate a sandwich .

(c) the president sighed .

(d) the president thought that a sandwich sighed .

(e) it perplexed the president that a sandwich ate Sally .

(f) the very very very perplexed president ate a sandwich .

(g) the president worked on every proposal on the desk .

(h) Sally is lazy .

(i) Sally is eating a sandwich .

(j) the president thought that sally is a sandwich .

You want to end up with a single grammar that can generate all of these sentences as well as grammatically similar ones. An important part of the problem is to generalize from the sentences above. For example, (b) is an invitation to think through the ways that conjunctions ("and", "or") can be used in English. (g) is an invitation to think about prepositional phrases ("on the desk","over the rainbow", "of the United States") and how they can be used.

While your new grammar may generate some very silly sentences, it should not generate any that are obviously ungrammatical. For example, your grammar must be able to generate (d) but not:

```
the president thought that a sandwich sighed a pickle .
```

Submit the CNF version of your new grammar in `grammar2-CNF.txt`. If you wrote a non-CNF grammar, please submit it as well in `grammar2.txt`.

(b) Think about the following phenomena, and extend your grammar to handle them:

- Relative clauses. Examples:
  - the pickle kissed the president that ate the sandwich .
  - the pickle kissed the sandwich that the president ate .
  - the pickle kissed the sandwich that the president thought that Sally ate .

  Hint: These sentences have something in common with (d) in the previous question.
- Singular vs. plural agreement. For this, you will need to use a present-tense verbs. Examples:
  - the citizens choose the president .

- the president chooses the chief of staff .
- the president and the chief of staff choose the sandwich .

Be sure you can handle the particular examples suggested, which means among other things your grammar must include the words in those examples. You should also generalize appropriately beyond these examples. Your final grammar should handle everything from the previous questions as well.

Submit the CNF version of your new grammar in `grammar3-CNF.txt`. If you wrote a non-CNF grammar, please submit it as well in `grammar3.txt`.

(c) Test your modified CNF grammar with the CKY parser (see full explanation at the beginning of this section). Submit some of your test results.