6.863J/9.611J: Natural Language Processing Prof. Robert C. Berwick

Assignment 2: Context-free grammar writing

My posse consists of: Sam Ainsley *Salman Ahmad (saahmad@mit.edu)*

1. **Hand in the output of a typical sample run of 10 sentences from your random sentence generator. Be sure to attach the code so we can test it.**

   ```
   is it true that the president kissed a chief of staff on a pickle on every chief of staff
   under a pickle in a pickle with the sandwich with every chief of staff on every floor
   ?
   ```

   ```
   a president under every chief of staff understood a pickle .
   ```

   ```
   a president on a delicious floor under the president under a sandwich under every pickle
   with every sandwich on every chief of staff under a president on the sandwich on the delicious
   pickle on every president on the pickle on a sandwich on a pickle under a chief of staff
   with a fine president under the chief of staff under the chief of staff on a president
   in every floor under the pickle in a delicious president with the pickle under a floor
   on a sandwich with a chief of staff on a sandwich in the chief of staff in the pickle
   under every floor with the pickle under every sandwich on the pickled pickled president
   in every sandwich in a sandwich with every floor on a perplexed sandwich on a floor on
   the chief of staff under a fine floor on a sandwich with every sandwich under every floor
   in the chief of staff under a floor in every pickle on a pickled delicious president under
   every perplexed chief of staff in the floor in a pickle with every pickle with the chief
   of staff on the chief of staff with a chief of staff with a floor under the floor in every
   floor with the floor with every chief of staff on every pickle in a chief of staff on
   every pickle under a chief of staff with a pickle on the president on a sandwich in the
   pickle with a pickle in a floor in a fine chief of staff under a sandwich under a pickle
   under the chief of staff under every chief of staff on every pickle with every president
   with the president on the sandwich in the president in the chief of staff in every president
   under a president on the pickle with the pickle in every fine president under a pickle
   on a pickled president with a sandwich on a president on the pickle with the fine pickle
   under the sandwich in the floor on every floor with every floor with every delicious floor
   in a sandwich under every chief of staff under every pickle under a sandwich in a pickle
   in every floor in every perplexed sandwich in a floor in every president on the pickle
   on every pickle in every chief of staff under a chief of staff in a sandwich under a fine
   sandwich with a pickle on every floor on a pickle under the delicious floor in every president
   under the floor under a president under the sandwich on every pickle with a chief of staff
   on the chief of staff in every president under the sandwich under the president under
   a sandwich with a sandwich on the president under the fine sandwich under a floor on a
   chief of staff in every sandwich with every chief of staff under every chief of staff
   with the perplexed sandwich with a sandwich under the fine president with every chief
   of staff on the floor on the chief of staff in the fine floor on every pickle under a
   sandwich under every president on every floor under the pickle in every floor in every
   pickled pickle on every president in every pickle on a president under every sandwich
   under every floor under every chief of staff with every chief of staff on every president
   under the fine floor under every chief of staff in the sandwich with every delicious floor
   under the president with a floor in every chief of staff on every sandwich with a president
   with a pickle in every president in the chief of staff on the chief of staff with a chief
   of staff under the sandwich under the pickled fine chief of staff on the pickle with the
   sandwich in a chief of staff pickled the pickle !  every president pickled a floor !
   ```

   ```
   the floor ate the chief of staff !
   ```

   ```
   every chief of staff in the pickle with every pickle in a perplexed sandwich on a pickle
   under every sandwich ate every pickle in the chief of staff under the sandwich on every
   ```

`floor in a sandwich with every president with every president with every chief of staff on the chief of staff in the floor in a sandwich under every sandwich on a floor on every sandwich on every delicious fine pickle on a floor in the president ! a floor ate every floor under every sandwich under every president with the pickle in the sandwich under the perplexed sandwich under the delicious delicious fine chief of staff !`

`is it true that every president kissed every pickle ?`

`every floor on every sandwich in the sandwich pickled a sandwich under the floor in the president in a sandwich with the pickled sandwich in every chief of staff on the president in a perplexed pickle in the president on every president on every floor with the floor .`

`the sandwich ate a president .`

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

   Long sentences are generated when you have a rule that (or s set of rules) that form a recursive loop. That is a rule A that either directly calls A itself or calls another rule that eventually calls A. In our grammar, one such rule is:

   `NP` $\rightarrow$ `NP PP`

   The other things that are special about this rule is that there are only two NP rules. These means that 50% of the time we will be performing a recursive loop.

   (b) The grammar allows multiple adjectives, as in, `the fine perplexed pickle`. **Why do your generated program's sentences exhibit this so rarely?**

   The rule that allows multiple adjectives is:

   `Noun` $\rightarrow$ `Adj Noun`

   The important thing about this rule is that there are many Noun alternatives which means that it will be recursing at a lower probability.

   (c) **Which numbers must you modify to fix the problems in 2(a) and 2(b), making the sentences shorter and the adjectives more frequent? (Check your answer by running your new generator and show that they work.)**

   There are many ways to achieve this. In my case, I modified the following rules:

   `2 NP` $\rightarrow$ `Det Noun`

   `20 Noun` $\rightarrow$ `Adj Noun`

   I could have also gotten the same results by making the following change instead of the first one above:

   `0.5 NP` $\rightarrow$ `NP PP`

   (d) **What other numeric adjustments can you make to the grammar in order to favor more natural sets of sentences? Experiment. Hand in your grammar file as a file named `grammar2`, with comments that motivate your changes, together with 10 sentences generated by the grammar.**

   The grammar file is included in my source submission. The generated sentences can be seen here:

   `every president understood a pickled chief of staff .`

   `the president under every fine president with a sandwich on a chief of staff in a floor in every sandwich with a pickle under the president under a president understood a pickled chief of staff on the chief of staff .`

   `the president wanted every president .`

   `is it true that every chief of staff wanted every sandwich ?`

   `a president on the pickle with the delicious president with every pickle with every chief of staff in every pickled president ate the president !`

   `every pickle pickled the president .`

   `every president under the fine pickled pickled chief of staff pickled every fine sandwich in every pickled president !`

```
is it true that a fine delicious president wanted the pickled pickle under every chief
of staff ?
every perplexed sandwich kissed the president on a chief of staff with a president
!
the fine chief of staff understood the president in the chief of staff .
```

3. **Modify the grammar into a new single grammar that can also generate the types of phenomena illustrated in the following sentences.**

   (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) `that a sandwich ate Sally perplexed the president .`

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

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

   Briefly discuss your modifications to the grammar. Hand in the new grammar (commented) as a file named `grammar3` and about 10 random sentences that illustrate your modifications.

   Below is a brief summary of my changes. Additional comments with more detail are available in my grammar file.

   Sentence (a)

   To support this sentence I needed to add proper nouns to the grammar. I am assuming that we don't have to deal with the case where a proper noun is preceded by a determiner (for example "the United States") so I enforced that determiners can allow precede non-proper nouns.

   Sentence (b)

   I added support for coordinating conjunctions - basically a conjunction that separates two phrases or clauses. The tricky thing here is that the conjunction must connect two phrases of the same type. For example, a NP and a NP not a NP and a VP.

   Sentence (c)

   I added support for intransitive verbs. This required me to break up the verb rule into both transitive and intransitive verbs as well.

   Sentence (d)

   I could not find the correct terminology for this type of verb, but I essentially added support for verbs that are followed by a "that" clause.

   This also made me add the word "that" as a demonstrative adjective.

   Sentence (e)

   To produce this sentence we need to support shifting the ordering of a sentence to allow "that clauses" to appear at the beginning. I believe this is an example of a deixis expression. Since it only happened in one place I did not try to abstract and generalize the rule.

   Sentence (f)

   I added support for intensifiers that modify adjectives. There is now a recursive rule that allows you to tack on as many modifiers to an adjective as possible.

   Sentence (g)

   I needed to add support phrasal verbs. Basically, verbs that are followed by a preposition that change the meaning. Note that this does not mean in general that any verb can be followed by a prepositional phrase.

   Other

   I also added any missing words to the vocabulary as was necessary.

4. **Give your program an option "-t" that makes it produce trees instead of strings. Generate about 5 more random sentences, in tree format. Submit them as well as the commented code for your program.**

   The generated sentences are:

```
(START (ROOT (S (NP (NP (NP (NP (Det the)
                                (Noun chief
                                      of
                                      staff))
                            (PP (Prep in)
                                (NP (Det the)
                                    (Noun floor))))
                        (PP (Prep in)
                            (NP (Det the)
                                (Noun (Adj pickled)
                                      (Noun president)))))
                    (PP (Prep with)
                        (NP (Det every)
                            (Noun chief
                                  of
                                  staff))))
                (VP (Verb ate)
                    (NP (Det a)
                        (Noun floor))))
            .))


(START (ROOT (S (NP (Det every)
                    (Noun floor))
                (VP (Verb ate)
                    (NP (Det every)
                        (Noun (Adj fine)
                              (Noun chief
                                    of
                                    staff)))))
            .))


(START (ROOT (S (NP (Det a)
                    (Noun president))
                (VP (Verb pickled)
                    (NP (Det the)
                        (Noun chief
                              of
                              staff))))
            !))


(START (ROOT (S (NP (NP (Det the)
                        (Noun chief
                              of
                              staff))
                    (PP (Prep in)
                        (NP (NP (Det every)
```

```
                            (Noun president))
                        (PP (Prep in)
                            (NP (NP (NP (Det every)
                                        (Noun pickle))
                                    (PP (Prep on)
                                        (NP (NP (Det the)
                                                (Noun (Adj delicious)
                                                      (Noun president)))
                                            (PP (Prep under)
                                                (NP (NP (Det the)
                                                        (Noun president))
                                                    (PP (Prep under)
                                                        (NP (Det a)
                                                            (Noun sandwich))))))))
                                (PP (Prep with)
                                    (NP (Det a)
                                        (Noun president))))))))
           (VP (Verb wanted)
               (NP (NP (Det a)
                       (Noun sandwich))
                   (PP (Prep with)
                       (NP (Det a)
                           (Noun floor))))))
       .))


(START (ROOT is
             it
             true
             that
             (S (NP (Det the)
                    (Noun pickle))
                (VP (Verb understood)
                    (NP (NP (Det a)
                            (Noun president))
                        (PP (Prep in)
                            (NP (Det every)
                                (Noun (Adj fine)
                                      (Noun chief
                                            of
                                            staff)))))))
             ?))
```

5. When I ran my sentence generator on `grammar`, it produced the sentence:

   `every sandwich with a pickle on the floor wanted a president .`

   This sentence is ambiguous according to the grammar, because it could have been derived in either of two ways.

   (a) One derivation is as follows; **what is the other one?**

```
(START (ROOT (S (NP (NP (NP (Det every)
```

```
                                   (Noun sandwich))
                                        (PP (Prep with)
                                            (NP (Det a)
                                                (Noun pickle))))
                                   (PP (Prep on)
                                       (NP (Det the)
                                           (Noun floor))))
                               (VP (Verb wanted)
                                   (NP (Det a)
                                       (Noun president))))
                    .))
```

The other derivation is:

```
(START (ROOT (S (NP (NP (NP (Det every)
                            (Noun sandwich))
                                 (PP (Prep with)
                                     (NP (Det a)
                                         (Noun pickle)
                                             (PP (Prep on)
                                                (NP (Det the)
                                                    (Noun floor)))))))
                               (VP (Verb wanted)
                                   (NP (Det a)
                                       (Noun president))))
                    .))
```

(b) **Is there any reason to care which derivation was used?**

The reason why we should care is that the meaning of the sentence fundamentally changes. In the first derivation the sandwiches were on the floor. The the second derivation, the sandwiches had a pickle that was on the floor.

6. (a) **Does the parser always recover the original derivation that was "intended" by `randsent`? Or does it ever "misunderstand" by finding an alternative derivation instead? Discuss. (This is the only part of question 6a that you need to answer.)**

It does "misunderstand" very often. The reason for this is because there is no one single valid parse tree for any given sentence (in general). The parse program stops after it finds a valid parse and reports just that one tree. It does not print out all of the trees (and it is possible for there to be many trees).

Long story short, there is not a 1-to-1 mapping between sentences and parse tree in our CFG. This is unlike, for example, the grammars and parsers for many programming languages where there almost always one (and only one) valid parse tree for a particular chunk of source code.

(b) **How many ways are there to analyze the following Noun Phrase (NP) under the original grammar? Explain your answer.**

There are a total of 5 possible parses. The trick to see this is that there are 3 total prepositions in this sentence. Whenever we see a new PP it can apply to any of the NP that came before it. So, when we see the first PP, it can only apply to the first NP. The second PP can apply to either of first 2 NP. And the last PP can apply to any of the previous 3. Thus, in total, we have $1 + 2 + 3 = 5$.

This can be confirmed by running the following command:

```
echo 'every sandwich with a pickle on the floor under the chief of staff' | ./parse -c -s NP -g grammar
```

(c) By mixing and matching the commands above, generate a bunch of sentences from `grammar`, and find out how many different parses they have. Some sentences will have more parses than others. **Do you notice any patterns? Try the same exercise with `grammar3`.**

You quickly notice an exponential blow up. Small sentences have a handful of parses, while larger sentences jump to crazy high numbers like 35357670.

These blowups are almost always the result of "left-recursive" rules. A "left-recursive" rule is follows this pattern: A → A X. The reason for this is that the X can apply to any of the preceding As. This leads to a blow up in the number of valid parses.

i. **Probability analysis of first sentence: Why is `p(best_parse)` so small? What probabilities were multiplied together to get its value of 5.144032922e-05?**
`p(sentence)` is the probability that `randsent` would generate this sentence. **Why is it equal to `p(best_parse)`?**
Why is `p(best_parse|sentence)`=1?
$p(best_parse)$ is so small because there are a huge number of possible sentences that can share this parse trees. The number that were multiplied are the **normalized** numbers in the first column of a grammar rule.
The probability is exactly 1 because there is only one valid parse tree for this particular sentence.

ii. **Probability analysis of the second sentence:**
What does it mean that `p(best_parse|sentence)` is 0.5 in this case?
Why would it be *exactly* 0.5?
The probability is exactly 0.5 because there are 2 parses for the sentence. Thus, the number of times that the parse will appear will be half of the number of times that the sentence itself appears.

iii. **Cross-entropy of the two sentence corpus. Explain exactly how the following numbers below were calculated from the two sets of numbers above, that is, from the parse of each of the two sentences.**
The general formula cross-entropy is not useful for us because the probability distribution is not known. Thus, we have to estimate the cross entropy based on the sentences that we can see from the set. It is shown in the equation below:

$$cross - entropy = -\sum_{i=0}^{n} \frac{1}{n} log_2(p(s_i)) \tag{1}$$

where s_i is the ith sentence in our corpus of n sentences. The actual math for our particular case is show below:

$$cross{-}entropy = -\frac{log_2(p(s_1)) \cdot log_2(p(s_2))}{size(vocab)} = -\frac{log_2(5.14 \times 10^{-}5) + log_2(1.24 \times 10^{-}9)}{18} = 2.43 \tag{2}$$

iv. **Based on the above numbers, what *perplexity* per word did the grammar achieve on this corpus?**
The perplexity is computed as $2^{cross-entropy}$. In this case it comes out to: 5.41.

v. **The compression program might not be able to compress the following corpus that consists of just two sentences very well. Why not? What cross-entropy does the grammar achieve this time? Try it and explain.** (The new 2 sentence corpus is given below.)
The cross-entropy in this case is ∞ because the second sentence is not in the grammar. Since the probability of finding that sentence is 0 the cross-entropy goes to ∞ because $log(0) = -\infty$ Since the sentence is not in the grammar, we have no way of using that grammar to compress the corpus. Thus, we will need (potentially) an infinite amount of bits so we can literally store each and every sentence that appears. This goes to show the power of leveraging structure in compression schemes.

vi. **How well does `grammar2` do on average at predicting word sequences that it generated itself? Please provide an answer in bits per word. State the command (a Unix pipe) that you used to compute your answer.**
The entropy for grammar2 seems to be around 2 bits. The command that I used to test this is:

`./randsent grammar2 10000 | /path/to/.../parse -C -g ./grammar2`
The key to getting a good representative value is to generate a large number of random sentences. In my case, I generated 10,000. During one run I generated the following output:
`cross-entropy = 2.03467 bits = -(-341154 log-prob.  / 167671 words)`

vii. If you generate a corpus from `grammar2`, then `grammar2` should on average predict this corpus better than `grammar` or `grammar3` would. In other words, the entropy will be *lower* than the cross-entropies. **Check whether this is true: compute the numbers and discuss.**

7. **Think about all of the following phenomena, and extend your grammar from question 3 to handle them. (Be sure to handle the particular examples suggested.) Call your resulting grammar `grammar4` and be sure to include it in your write-up along with examples of it in action on new sentences like the ones illustrated below.**

   I have included sample outputs from the grammar below. I also tested to ensure all of the sentences and test case presented in the assignment pass. You can see the test case file in `test.rb` in my source distribution. You can easily see if everything passes by running:

   `./test.rb | grep failure`

   (a) *Yes-no questions.*

   `did Sally eat the sandwich with Sally ?`

   `will Sally eat that a president and a very very fine very pickled very very perplexed very very very pickled sandwich ate ?`

   `will every pickle think that the Sally worked on Sally and Sally ?`

   `did Sally with Sally and Sally and the perplexed Sally think that Sally worked and worked ?`

   `will Sally in the who think the president under Sally in a desk and Sally in the very perplexed pickle and the desk and every perplexed Sally under Sally on every pickled Sally and Sally ?`

   (b) *WH-word questions.*

   `who worked and worked on Sally ?`

   `where did Sally eat ?`

   `who understood that a president in a Sally thought that Sally and Sally under Sally under Sally under a Sally and Sally and every Sally and Sally under Sally on Sally on a perplexed Sally and a fine Sally in the fine Sally and a very very pickled perplexed Sally on a Sally and Sally and every very very very very very very perplexed Sally in Sally on Sally on Sally under every pickle and Sally and Sally in a who and a desk on Sally and the who with every floor and Sally on every very delicious who in Sally in the very very fine Sally and Sally and every very very very very fine perplexed Sally and a who in Sally and the who with Sally and the who and a who and Sally and the floor on the very delicious Sally on a delicious chief of staff kissed Sally with Sally and Sally ?`

   `what did Sally and Sally think that Sally in Sally worked with Sally and every Sally under ?`

   `who thought that the floor ate and worked under Sally ?`

   `where did Sally eat that Sally and the delicious delicious fine who and Sally in every delicious Sally and a who in every very very delicious Sally and Sally and the Sally with Sally thought that the who sighed and ate ?`