Massachvsetts Institvte of Technology
Department of Electrical Engineering and Computer Science
6.863J/9.611J, Natural Language Processing
Warmup for Laboratory 2: Designing Context-Free Grammars

Name: *Conleigh Byers*
KerberosID: *conleigh*
Collaborators:

---

1 `randsent`, a random sentence generator

Problem 1. Show the output produced when you run `./randsent grammar 10` (that is, 10 sentences randomly generated from the grammar in the file named `grammar`). ◁

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

2 Weighted generation

Problem 2. Why does `randsent` generate such long sentences from `grammar`? Which grammar rule is responsible, and why? What distinguishes this rule from the other rules in the grammar? ◁

The script generates such long sentences because sometimes it will get caught in loops of generating more nonterminals that are not preterminals by following a rule that has as its right-hand side other nonterminals that are not preterminals. The rule 1 NP NP PP seems particularly guilty, as it can lead to expansion of phrases quite quickly. It is the only rule that has as part of its right-hand side its left-hand side, i.e., it rewrites as itself (a nonterminal) plus another nonterminal, where the LHS in question is not a preterminal.

Problem 3. The grammar admits consecutive adjectives, as in `the fine perplexed pickle`. Why do consecutive adjectives occur so rarely? ◁

Consecutive adjectives occur so rarely because the only way for an adjective to occur is if the rule '1 Noun Adj Noun' is selected, but this rule is selected with probability equal to selecting the rules '1 Noun president', '1 Noun sandwich', etc. It is thus even rarer that this rule of a Noun becoming Adj Noun would be selected twice in a row, which is what must happen for consecutive adjectives to occur.

Problem 4. Which rules' weights must you modify to fix the issues identified in problem 2 and problem 3, and what are the new weights? Make these changes in `grammar2`. (You can check your answer by running `randsent`.) ◁

To fix the issue of long sentences, we can decrease the weight of the rule 'NP NP PP' relative to other rules with LHS 'NP'. Let's change the weight of this rule to .2 but leave the other 'NP' rule as 1. To increase the amount of times that consecutive adjectives appear, we can increase its weight; let's change it to 4.

The results of these changes is the following sample output:

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

Problem 5. What other numeric adjustments can you make to `grammar2` to favor more natural sentences? Explain the changes you made. Then provide 10 example sentences generated from `grammar2`. ◁

The prior question requested an increase in the number of consecutive adjectives, although I'm not sure I'd classify that as more natural-sounding. Let's change the weight of 'N Adj N' to 2. Syntactically, these sentences seem reasonably natural. Semantically, of course, they do not.

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

3 Trees

Problem 6. `grammar` can generate 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. One derivation is given below. What is the other derivation? Is there any reason to care which derivation is used? ◁

First, we document that trees can be generated and are working as expected.

```
every pickled sandwich kissed the pickle !
(START (ROOT (S (NP (Det every) (Noun (Adj pickled) (Noun sandwich))) (VP (Verb kissed) (NP (Det the) (Noun pickle)))) !))
the pickle wanted every sandwich !
(START (ROOT (S (NP (Det the) (Noun pickle)) (VP (Verb wanted) (NP (Det every) (Noun sandwich)))) !))
is it true that the president pickled every floor ?
(START (ROOT (is) it true that (S (NP (Det the) (Noun president)) (VP (Verb pickled) (NP (Det every) (Noun floor)))) ?))
a sandwich kissed every sandwich .
(START (ROOT (S (NP (Det a) (Noun sandwich)) (VP (Verb kissed) (NP (Det every) (Noun sandwich)))) .))
```

And a check that piping through prettyprint yields a suitable output:

```
(START (ROOT (is )
             it
             true
             that
             (S (NP (Det a)
                    (Noun floor))
                (VP (Verb wanted)
                    (NP (Det every)
                        (Noun president))))
             ?))
```

All seems well.

For the sentence at hand, 'every sandwich with a pickle on the floor wanted a president', there is an alternate derivation. One derivation means the pickle is on the floor, while one means the sandwich is on the floor. This alternative derivation is given below in tree form.

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

As to whether there is reason to care which derivation is used, the derivation matters if we care about meaning. The two sentences do not convey the same state of affairs in regards to our presidentially-minded sandwich.

4 New grammar phenomena

Problem 7. Briefly discuss the modifications you made to `grammar3` to support generation of sentences similar to those above. Provide 10 sentences generated from your modified grammar. ◁

The grammar3 file is able to pass all 10 test cases after some modifications. Complement clauses, complementizers, conjunctions, and distinctions between verbs that are intransitive, transitive, and those that may precede a complementizer are introduced. A special case verb category is added to handle constructions of the form of sentence 8 in the grammar. As a native English speaker, there seems to be something special about the kind of verb that may come after the complement cause beginning a sentence beyond simply being a transitive verb. For this example, I have left it as transitive verbs, but in reality I suspect something more subtle is going on.

```
is it true that every president kissed on sandwich but the very proposal kissed every proposal ?
proposal sighed in every pickle and proposal on a president with every desk in proposal !
delicious very perplexed Sally ate a chief of staff in the president !
every floor and Sally kissed !
every proposal worked .
is it true that proposal under pickle  perplexed president ?
is it true that chief of staff in the delicious pickle and president and every proposal on chief of staff understood a pickle ?
a sandwich wanted the desk .
a proposal kissed !
the floor with the sandwich kissed desk .
```

5 `parse`, a parser

5.1 Ambiguity

Problem 8. Use `randsent -t grammar3` to generate random sentences from `grammar3`, then try parsing those same sentences (as strings, not trees) using `parse -g grammar3`. Does the parser always recover the original derivation that `randsent` used when generating the sentence? Explain why or why not. ◁

I tested several sentences that should have multiple possible derivations, but when running parse -g grammar3, I only get 1 of the possible derivations each time, and it matches the tree structure output by my own randsent program. The sentences I tested included the example from above, 'every sandwich with a pickle on the floor wanted a president .' as well as the randomly generated sentences 'the chief of staff sighed with every perplexed sandwich on the floor .' and 'pickle understood that Sally worked under floor under desk on sandwich but desk wanted very fine proposal on pickle !' All have ambiguity in regards to what subject the secondary prepositional phrase is attached to. However, each time I get the same derivation from my own randsent program and from parse, and these derivations are the same.

Section 5.2 sheds some light on why I'm seeing this behavior, as parse will return the most probable parse by default. Thus, I shouldn't expect to see different derivations each time running parse without additional flags. I think my getting the same derivation as parse's most probable may be coincidental; my program may expand nonterminals in such a way that it is what is usually the most probable, by whatever criteria parse is using.

Problem 9. How many ways are there to parse the following NP under the original `grammar`? (That is, how many ways are there to derive the following string starting from the NP nonterminal symbol?)

```
every sandwich with a pickle on the floor under the chief of staff
```

Explain your answer. Then show the `parse` command line you can use to check your answer. (Hint: run `parse -h` to see all the flags supported by `parse`.) ◁

First, this string fails to parse if entered in as-is with the original grammar file. If we add both a VP and a period, the sentence will parse.

The new sentence:

```
every sandwich with a pickle on the floor under the chief of staff ate a pickle .
```

yields 5 different parses. We can find this by running parse with the flag -c.

```
6.863$ /mit/6.863/spring2018/Software/parse -g grammar -c
a president pickled every chief of staff !
(START (ROOT (S (NP (Det a) (Noun president)) (VP (Verb pickled) (NP (Det every) (Noun chief of staff)))) !))
        no._parses= 1
every sandwich with a pickle on the floor under the chief of staff ate a pickle .
(START (ROOT (S (NP (NP (NP (Det every) (Noun sandwich)) (PP (Prep with) (NP (Det a) (Noun pickle)))) (PP (Prep on) (NP (NP (D
et the) (Noun floor)) (PP (Prep under) (NP (Det the) (Noun chief of staff))))))) (VP (Verb ate) (NP (Det a) (Noun pickle)))) .)
)
        no._parses= 5
```

This seems right. There are three prepositional phrases at the bottom of the tree. Let's call them PP1 = 'with a pickle', PP2 = 'on the floor', PP3 = 'under the chief of staff'. Possible derivations include:
'every (sandwich that is PP1, PP2, and PP3)'
'every (sandwich that is PP1) is PP2 and PP3'
'every (sandwich that is PP1 and PP2) is PP3'
'every (sandwich that is PP1) is (PP2 that is PP3)'
'every (sandwich) that is (PP1 that is PP2 that is PP3)'

Problem 10. Generate sentences from `grammar` and check how many different parses they have. Which kinds of sentences have more parses than others, and why? Now generate sentences from `grammar3`. Again, which kinds of sentences have more parses than others, and why? ◁

Generated sentences and number of parses are shown below for grammar and grammar3. For grammar, the number of parses to a sentence can get so large the program crashes. Shown below is a large example (4862 parses) from a sentence with a lot of embedded prepositional phrases, which have many derivational options, as demonstrated in the prior question. For grammar3, the sentences are overall shorter and number of parses is much smaller. However, parses can increase quite a bit when conjunctions and embedded prepositional phrases are involved.

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

(START (ROOT is it true that (S (NP (Det every) (Noun chief of staff)) (VP (Verb pickled) (NP (Det every
no._parses= 1
(START (ROOT is it true that (S (NP (Det a) (Noun (Adj fine) (Noun chief of staff))) (VP (Verb wanted)
no._parses= 1
(START (ROOT is it true that (S (NP (Det every) (Noun chief of staff)) (VP (Verb understood) (NP (NP (D
no._parses= 1
(START (ROOT (S (NP (NP (NP (NP (NP (Det every) (Noun floor)) (PP (Prep with) (NP (Det a) (Noun preside
no._parses= 4862

a desk understood every president !
the chief of staff wanted perplexed president .
is it true that every chief of staff sighed ?
desk wanted sandwich .
a pickled fine president worked .
president thought on proposal on pickled proposal .
is it true that every floor sighed but sandwich sighed ?
every Sally understood that desk under pickle and sandwich kissed on the Sally but very chief of staff
floor sighed !
```

```
the sandwich ate every proposal .

(START (ROOT (S (NP (Det a) (Noun desk)) (VP (Verb understood) (NP (Det every) (Noun president)))) !))
no._parses= 1
(START (ROOT (S (NP (Det the) (Noun chief of staff)) (VP (Verb wanted) (NP (Noun (Adj perplexed) (Noun
no._parses= 1
(START (ROOT is it true (CC (Comp that) (S (NP (Det every) (Noun chief of staff)) (VP (Verbi sighed))))
no._parses= 1
(START (ROOT (S (NP (Noun desk)) (VP (Verb wanted) (NP (Noun sandwich)))) .))
no._parses= 1
(START (ROOT (S (NP (Det a) (Noun (Adj pickled) (Noun (Adj fine) (Noun president)))) (VP (Verbi worked)
no._parses= 1
(START (ROOT (S (NP (Noun president)) (VP (VP (VP (Verbi thought)) (PP (Prep on) (NP (Noun proposal))))
no._parses= 2
(START (ROOT is it true (CC (Comp that) (S (NP (Det every) (Noun floor)) (VP (Verbi sighed)) (Conj but)
no._parses= 1
(START (ROOT (S (NP (Det every) (Noun Sally)) (VP (Verbc understood) (CC (Comp that) (S (NP (NP (NP (Nou
no._parses= 6
(START (ROOT (S (NP (Noun floor)) (VP (Verbi sighed))) !))
no._parses= 1
```

5.2 Probability

Problem 11. For the first sentence, `parse` reports:

```
p(sentence)= 5.144032922e-05
p(best_parse)= 5.144032922e-05
p(best_parse|sentence)= 1
```

Why is `p(sentence)` so small? What probabilities were multiplied together to get its value? Why does `p(best_parse) = p(sentence)`? Why does `p(best_parse|sentence)` = 1? ◁

We can compute the probability of a given parse (derivation) of a sentence by multiplying the probabilities of each of the rules used in the derivation. p(sentence) is so small because there are so many possible sentences that could be generated from this grammar. p(sentence)=p(best_parse) because there is only one possible derivation of this sentence.

p(best_parse—sentence) = 1 because there is only one possible derivation of the first sentence, so the probability of the best parse given that we see this sentence is 100%.

Problem 12. For the second sentence, `parse` reports:

```
p(sentence)= 1.240362877e-09
p(best_parse)= 6.201814383e-10
p(best_parse|sentence)= 0.5
```

What does the value of `p(best_parse|sentence)` mean? Why is it *exactly* 0.5? ◁

p(best_parse—sentence) = 0.5 exactly because there are 2 possible derivations (parses) for the sentence, and presumably their probabilities of occurring are equal. So, the conditional probability that we receive the 'best' parse given that we saw this sentence is 50%.

p(sentence) is presumably the sum of the probabilities of all possible derivations, one of which is the most probable, or the 'best' parse.

5.3 Compression

Problem 13. Explain exactly how each of the numbers in the cross-entropy line printed by `parse` can be computed from the probabilities listed in problems 11 and 12. ◁

The probabilities of each of the two sentences in the corpus are p1 = 5.144032922e-05 and p2 = 1.240362877e-09. We can find the cross-entropy found above by the formula $2.435bits = -(1/18)*(log_2(5.144032922e-05) + log_2(1.240362877e-09))$.

Problem 14. Here is another corpus.

```
the president ate the sandwich .
the president ate .
```

What cross-entropy does `grammar` achieve on this corpus? Explain why. ◁

The command fails with output 'cross-entropy = inf bits = -(-inf log-prob. / 10 words)'. This is because grammar has no intransitive verbs. All verbs have objects in the grammar file. Since 'the president ate .' cannot be explained by the grammar, the compression scheme with reference to grammar cannot represent the new corpus with a finite number of bits per symbol. We can confirm this by modifying the corpus such that the second sentence says 'the president ate the president .' Now we get the output 'cross-entropy = 2.37446 bits = -(-28.4935 log-prob. / 12 words)', as expected.

Problem 15. Generate a corpus of 1000 sentences from `grammar2` and find the cross-entropy (in bits per symbol) of this corpus against `grammar2` itself. Show the command line you used to generate this value. (It should contain a `|` (the pipe character).) ◁

Using the commands './randsent grammar2 1000 ¿ g2-1000.txt' and 'parse -g grammar2 -C -i g2-1000.txt' yields the output 'cross-entropy = 1.96238 bits = -(-21119.1 log-prob. / 10762 words)'. Alternatively, we could use the two commands with a pipe instead of creating the text files first.

Problem 16. Estimate the entropy of `grammar3`. How does it compare to the entropy of `grammar2`? Explain why. ◁

Using the command 'parse -g grammar3 -C -i g3-1000.txt' yields the output 'cross-entropy = 3.1158 bits = -(-35588.6 log-prob. / 11422 words)'. The entropy of grammar3 is larger than of grammar2. Presumably this is because grammar3 has more rules and longer possible sentences.

Problem 17. Approximate (using a corpus of 1000 sentences) the cross-entropy of `grammar2` against `grammar`. How does this cross-entropy compare to `grammar2`'s entropy? Explain why. ◁

Using the command 'parse -g grammar -C -i g2-1000.txt' yields the output 'cross-entropy = 2.08629 bits = -(-22452.7 log-prob. / 10762 words)'.

This cross-entropy is close to but a little bigger than the entropy of grammar 2. This makes sense, as grammar2 should be better at representing a corpus of sentences generated from grammar2 than grammar.

Problem 18. Now approximate (using a corpus of 1000 sentences) the cross-entropy of `grammar2` against `grammar3`. How does this cross-entropy compare to `grammar2`'s entropy? Explain why. ◁

Using the command 'parse -g grammar3 -C -i g2-1000.txt' yields the output 'cross-entropy = 2.64752 bits = -(-28492.7 log-prob. / 10762 words)'. This cross-entropy is also bigger than grammar2's entropy. This is because grammar2 is better at representing sentences generated from grammar2 than grammar3 is - the probabilities more closely align, and less rules are required. Grammar3 has rules that are not needed to generate all possible output from grammar2.

6 More new grammar phenomena

Problem 19. Briefly discuss the modifications you made to the grammar to support these forms of questions. Provide 10 sentences generated from your modified grammar. ◁

To support these forms of questions, I added several different kinds of questions. There was some overlap, but some distinctions in the kind of syntax possible. For instance, 'where' was separated from 'who' and 'what'. In one category, we have the syntax of the 'did'/'will' questions, which may be preceded by 'where'. In the other category are questions generated from 'who'/'what', which have a modified grammar to take into account that 'who' or 'what' can be the object of a transitive verb or the object of a prepositional phrase. Several new types of verb categories are added: infinitive verbs that are transitive, intransitive, and preceding a complementizer. With these modifications, the grammar4 file passes all tests seen so far.

```
who will the perplexed president eat ?
Sally thought that a very chief of staff kissed proposal on the proposal !
every president  perplexed fine sandwich !
where did a chief of staff understand that every chief of staff kissed ?
where will every sandwich perplex every delicious pickled floor ?
is it true that the floor worked ?
every desk worked but the desk sighed .
did floor perplex the very sandwich ?
is it true that every sandwich understood that the chief of staff kissed every chief of staff in the Sally with a desk with every chief of staff ?
who did the Sally work on ?
```