Siwei Bian, Shreayan Chaudhary, Kevin Li
September 14, 2022
Homework #1 Natural Language Processing 601.665

1.4 Questions

1.
    a. a pickle kissed the chief of staff .
    b. a sandwich on every chief of staff on a pickle under a delicious chief of staff wanted every floor !
    c. is it true that every president with a floor under a president in the sandwich under a chief of staff kissed a delicious floor with a sandwich under a president ?
    d. a president kissed a pickled floor !
    e. is it true that every perplexed sandwich pickled every sandwich under the pickle on every president in every pickle with a sandwich under a chief of staff in the floor in every fine floor in every sandwich with a pickle under a chief of staff in the floor under the pickle under a floor with a sandwich on the president under the president in every president in the floor under the president in the president with a chief of staff in a chief of staff under every floor under a sandwich under the president in the floor with a sandwich with every president in the pickle under a chief of staff with the delicious president on a pickle under every sandwich with the chief of staff on every president on a pickle with a president with a delicious pickle on every pickle with a perplexed pickle under every sandwich with a pickle under every pickle in the president on a sandwich with every sandwich in the delicious sandwich with the sandwich ?
    f. a fine president pickled the president in a chief of staff !
    g. the chief of staff in the sandwich with a chief of staff in the sandwich with a president under every chief of staff in every floor in every floor pickled the delicious fine chief of staff with the delicious chief of staff with the floor on a chief of staff on every delicious perplexed sandwich .
    h. the pickle with the president pickled every chief of staff .
    i. a sandwich wanted the chief of staff !
    j. the floor understood the floor .

2.
    a.
```
(ROOT is
      it
      true
      that
      (S (NP (Det every)
             (Noun sandwich))
      (VP (Verb ate)
             (NP (Det a)
                    (Noun president))))
      ?)
```
    b.
```
(ROOT (S (NP (NP (NP (Det every)
```

```
                        (Noun pickle))
                        (PP (Prep under)
                        (NP (Det a)
                              (Noun (Adj fine)
                                    (Noun floor)))))
                  (PP (Prep under)
                        (NP (Det every)
                        (Noun sandwich))))
            (VP (Verb understood)
                  (NP (Det the)
                        (Noun (Adj perplexed)
                              (Noun president)))))
            .)
```

3.

```
      a. (ROOT (S (NP (NP (NP ......)
                        ...)
                  ...)
            ...)
            ...)
      b. (ROOT is
            it
            true
            that
            (S (NP (NP (Det the)
                        ...)
                  ...)
            ...)
            ...)
```

2.1 Questions

1. The only ways for the program to reach terminal noun symbols are through the `NP → Det Noun` and `Noun → Adj Noun` rules. This often leads to the creation of long sentences because, not only is there only a ½ chance that a given `NP` expansion will lead to the `Noun → Adj Noun` rule, but the other ½ of the time, the `NP` expansion recurses and leads to another expansion of an `NP` and `PP`. In addition, the rules with `VP` and `PP` on the LHS require an expansion of `NP`. This means there is a high number of expansions which, more likely than not, will result in a nonterminal symbol, leading to a long sentence.
2. In order for there to be multiple adjectives in a noun phrase, the `Noun → Adj Noun` rule must be used in consecutive expansions. Since each terminal noun and the above rule are weighted equally, there is only a ⅙ chance after `Noun` appears in the `NP → Det Noun` rule that this rule will be used in the first place (one adjective), and then a ⅙ chance that the rule will be used immediately afterwards (two adjectives). So, there is a small (1 in 36) chance of having even a short chain of adjectives.
3. In order to have shorter sentences, we want to get to terminal symbols more quickly. We can achieve this by weighting the `NP → Det Noun` rule higher relative to the other rules for `NP` expansions. In addition, to make adjectives more frequent, we can weight the `Noun → Adj Noun` rule higher relative to the other rules for `Noun` expansions. In this case, we assign a weight of 3 to the `NP → Det Noun` rule and a weight of 8 to the `Noun → Adj Noun` rule (grammar2.gr).
4. One "unnatural" phenomenon in the generated sentences using the above weights was that sentences occasionally appeared with too many consecutive `NP → NP PP` expansions such as "`... kissed the floor in a sandwich under every delicious chief of staff !`" To rectify this, we can even further bias the weights to select this rule less frequently by assigning a higher weight to the only other `NP` rule, `NP → Det Noun`. This leads to generation of sentences with fewer consecutive prepositional noun phrases.
5.
    a. `is it true that the pickled delicious pickled pickled floor wanted every chief of staff ?`
    b. `a president ate every chief of staff !`
    c. `the perplexed pickle ate the delicious pickle !`
    d. `is it true that every pickled pickled sandwich understood a fine pickle ?`
    e. `the delicious president ate every fine delicious pickle in a floor .`
    f. `the fine perplexed pickled pickled delicious floor ate every floor !`
    g. `a delicious floor pickled every sandwich !`
    h. `the fine fine chief of staff pickled the fine perplexed floor !`
    i. `a chief of staff kissed the sandwich !`
    j. `every fine delicious chief of staff wanted every delicious pickled fine sandwich .`

2.3 Questions continued

1. In order to achieve the desired outputs, we made the following major changes:
    a. Define noun clause as the word *that* followed by S :
    ```
    NC   that S
    ```
    b. Define noun clause verbs (NCV) that usually precede a noun clause:
    ```
    NCV   thought
    NCV   said
    NCV   understood
    ```
    c. Define three kinds of composite sentences (CS) that contain noun clauses and noun clause verbs:
    ```
    CS   NP NCV NC
    CS   NC VP
    CS   it VP NC
    ```
    d. Define verbs to be in two categories: transitive verbs(TV) and intransitive verbs(ITV):
    ```
    TV  ate
    TV  wanted
    TV  kissed
    TV  understood
    TV  pickled

    ITV sighed
    ITV worked
    ```
    e. Replace the original VP expansion with:
    ```
    VP  TV NP (transitive verb followed by noun phrase)
    VP  ITV (intransitive verb by itself)
    VP  ITV PP (intransitive verb followed by a
    preposition)
    VP  VP and VP
    VP  TV and VP
    ```
    f. Define proper noun (Pnoun) which is a noun that doesn't need to follow a determiner:
    ```
    PNoun  Sally
    PNoun  JHU
    …… .
    ```
    g. To incorporate the word "very" to modify adjectives, we add a new Adj expansion:
    ```
    Adj   very Adj
    ```

2.
```
1)  that the sandwich in a floor worked on Sally sighed .
2)  every floor understood that Tom worked under JHU and
pickled a chief of staff on the chief of staff !
3) is it true that the delicious chief of staff sighed ?
4) Tom worked and worked !
5) a delicious chief of staff and Tom thought that a pickle
under a fine pickled floor perplexed the chief of staff !
6) that JHU sighed and understood a pickle in JHU worked !
```

7) it sighed under a president that Sally kissed the floor and Tom .
8) it worked that a sandwich and Sally sighed under JHU .
9) a sandwich understood that Tom sighed on a pickle and Tom .
10) it sighed in every floor that every floor worked .

3.1 Questions

a. Another possible derivation of "`every sandwich with a pickle on the floor wanted a president .`" is as follows:

```
(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))))
      .)
```

b. For the given sentence, though there are two possible derivations using our defined grammar, only the first derivation (the one given in the homework) is plausible if we consider each of the derivations' meanings. The first derivation tells us that, on the floor, there were sandwiches with pickles in them, and that every one of these sandwiches wanted a president. The second possible derivation, above, tells us that there were sandwiches and that these sandwiches, either along with or utilizing (different possible interpretations of "with") a singular pickle on the floor, wanted a president. Neither of the meanings implied by the second derivation are as semantically logical as the first. We see that the differing derivations create real distinctions in meanings of the same sentence, so we should care that we use the first derivation.

3.3 Questions continued

2. Using `grammar3.gr`, we generated the following sentences:
   1) `that every floor and Sally perplexed Tom sighed and worked on every fine pickle and sighed with a floor and ate a floor !`

   Derivation of the sentence:
```
(ROOT (CS (SC that
              (S (NP (NP (Det every)
                             (Noun floor))
                         and
                         (NP (PNoun Sally)))
                     (VP (TV perplexed)
                         (NP (PNoun Tom)))))
          (VP (VP (ITV sighed))
              and
              (VP (VP (VP (ITV worked)
                              (PP (Prep on)
                                  (NP (Det every)
                                      (Noun (Adj fine)
                                             (Noun pickle)))))
                          and
                          (VP (ITV sighed)
                              (PP (Prep with)
                                  (NP (Det a)
                                      (Noun floor)))))
                  and
                  (VP (TV ate)
                      (NP (Det a)
                          (Noun floor))))))
      !)
```

   2) `every sandwich kissed every pickle !`

   Derivation of the sentence:
```
(ROOT (S (NP (Det every)
             (Noun sandwich))
         (VP (TV kissed)
             (NP (Det every)
                 (Noun pickle))))
      !)
```

   3) `that every chief of staff with Sally sighed on Tom in JHU Tom worked in Sally !`

   Derivation of the sentence:
```
(ROOT (CS (SC that
              (S (NP (NP (Det every)
```

```
                        (Noun chief
                              of
                              staff))
                    (PP (Prep with)
                        (NP (PNoun Sally))))
                (VP (ITV sighed)
                    (PP (Prep on)
                        (NP (NP (PNoun Tom))
                            (PP (Prep in)
                                (NP (NP (PNoun JHU))
                                    and
                                    (NP (PNoun Tom)))))))))
         (VP (ITV worked)
             (PP (Prep in)
                 (NP (PNoun Sally)))))
    !)
```

4) the sandwich understood that every president with the pickle wanted Tom in Tom and wanted JHU !

Derivation of the sentence:

```
(ROOT (CS (NP (Det the)
              (Noun sandwich))
          (SCV understood)
          (SC that
              (S (NP (NP (Det every)
                         (Noun president))
                     (PP (Prep with)
                         (NP (Det the)
                             (Noun pickle))))
                 (VP (VP (TV wanted)
                         (NP (NP (PNoun Tom))
                             (PP (Prep in)
                                 (NP (PNoun Tom)))))
                     and
                     (VP (TV wanted)
                         (NP (PNoun JHU)))))))
    !)
```

5) that Tom ate Tom and ate the president sighed on every floor .

Derivation of the sentence:

```
(ROOT (CS (SC that
              (S (NP (PNoun Tom))
                 (VP (VP (TV ate)
                         (NP (PNoun Tom)))
                     and
                     (VP (TV ate)
```

```
                             (NP (Det the)
                                 (Noun president))))))
              (VP (ITV sighed)
                  (PP (Prep on)
                      (NP (Det every)
                          (Noun floor)))))
          .)
```

No, the parser does not always recover the original derivation 'intended' by randsent. In many cases, there are multiple ways to parse and understand the clauses in the sentence.
Example: Consider the sentence - every sandwich with a pickle on the floor wanted a president .
If we send the above sentence to the parser, it is not able to find out the intended meaning of the sentence. There are multiple ways to parse the sentence and it is not able to find the meaning.
One of the parsed meaning will be:

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

3. The given sentence can be parsed in 5 ways.

1. [every [sandwich [with a pickle on the floor]] [under the chief of staff]]
2. [[every [sandwich with a pickle] on the floor] [under the chief of staff]]
3. [every [sandwich with [a pickle on the floor]] [under the chief of staff]]
4. [every sandwich with [a pickle on the floor under the chief of staff]]
5. [every [sandwich with a pickle] [on the floor under the chief of staff]]

We ran the parse script to test out if we parsed the given phrase using the following command:

```
perl .\parse -g .\grammar.gr -s NP -c
every sandwich with a pickle on the floor under the chief of staff
 (NP (NP (NP (Det every) (Noun sandwich)) (PP (Prep with) (NP (Det a)
(Noun pickle)))) (PP (Prep on) (NP (NP (Det the) (Noun floor)) (PP
(Prep under) (NP (Det the) (Noun chief of staff
))))))
# number of parses = 5
```

The number of parses from the parser output matches the number of parses we have done manually.

4. Command:

```
python randsent.py -g grammar.gr -n 5 | perl .\parse -g grammar.gr -c
```

Sentences generated using grammar.gr:

```
   1.  (ROOT (S (NP (PNoun Sally)) (VP (ITV sighed) (PP (Prep on) (NP
       (NP (NP (NP (NP (PNoun Tom)) and (NP (Det the) (Noun
      president))) and (NP (NP (PNoun JHU)) (PP (Prep on) (NP (NP
       (PNoun
      Sally)) and (NP (PNoun JHU)))))) and (NP (PNoun JHU))) (PP
       (Prep on) (NP (PNoun JHU))))))) .)
# number of parses = 132
   2.  (ROOT (S (NP (PNoun JHU)) (VP (VP (VP (VP (VP (ITV worked) (PP
       (Prep under) (NP (PNoun Tom)))) and (VP (ITV worked) (PP (Prep
      on) (NP (NP (Det a) (Noun sandwich)) (PP (Prep on) (NP (N
      P (PNoun Tom)) and (NP (Det a) (Noun chief of staff)))))))) and
       (VP (ITV worked) (PP (Prep under) (NP (Det every) (Noun
      pickle))))) and (VP (ITV worked))) and (VP (ITV worked)))) .)
# number of parses = 28
   3.  (ROOT (S (NP (Det the) (Noun chief of staff)) (VP (Verb
      kissed) (NP (NP (Det the) (Noun pickle)) (PP (Prep with) (NP
       (NP (Det every) (Noun pickle)) (PP (Prep on) (NP (NP (NP (NP
       (NP (
      Det every) (Noun (Adj perplexed) (Noun floor))) (PP (Prep
      under) (NP (NP (Det the) (Noun president)) (PP (Prep under) (NP
       (Det a) (Noun president)))))) (PP (Prep in) (NP (Det a) (Noun
      president)))) (PP (Prep in) (NP (Det the) (Noun sandwich))))
       (PP (Prep on) (NP (Det a) (Noun sandwich))))))))) .)
# number of parses = 429
   4.  (ROOT (S (NP (NP (Det a) (Noun chief of staff)) (PP (Prep on)
       (NP (Det every) (Noun president)))) (VP (Verb ate) (NP (NP (Det
      a) (Noun chief of staff)) (PP (Prep on) (NP (Det a) (Noun
      sandwich)))))) !)
# number of parses = 4
   5.  (ROOT (S (NP (Det the) (Noun pickle)) (VP (Verb ate) (NP (Det
      every) (Noun pickle)))) !)
# number of parses = 1
```

Sentences generated using grammar3.gr:

```
   1.  (ROOT (S (NP (Det every) (Noun (Adj fine) (Noun floor))) (VP
       (ITV sighed))) !)
# number of parses = 1
   2.  (ROOT (S (NP (PNoun JHU)) (VP (TV kissed) (NP (Det the) (Noun
      chief of staff)))) .)
# number of parses = 1
   3.  (ROOT (CS (SC that (S (NP (Det every) (Noun president)) (VP
       (VP (ITV sighed) (PP (Prep under) (NP (Det every) (Noun
      president)))) and (VP (ITV worked))))) (VP (VP (ITV worked))
      and (V
```

```
    P (VP (ITV worked) (PP (Prep in) (NP (Det every) (Noun
    pickle)))) and (VP (ITV sighed) (PP (Prep on) (NP (PNoun
    Tom))))))) .)
# number of parses = 2
  4.  (ROOT (CS it (VP (TV kissed) (NP (PNoun Sally))) (SC that (S
     (NP (Det the) (Noun pickle)) (VP (TV kissed) (NP (PNoun
    Sally)))))) .)
# number of parses = 1
  5.  (ROOT is it true that (S (NP (PNoun Tom)) (VP (ITV sighed) (PP
     (Prep under) (NP (NP (Det the) (Noun chief of staff)) (PP (Prep
    on) (NP (NP (PNoun Tom)) and (NP (PNoun JHU)))))))) ?)
# number of parses = 2
```

There is a pattern that forms when changing from grammar.gr to grammar3.gr:
The sentences are much shorter in length. As a result, there are fewer combinations of the grammar rules. Also, in the grammar3.gr, the rules are much more stricter and specific to construct grammatical sentences. As a result, the average number of parses per sentence is much lower for grammar3.gr as compared to grammar.gr.

5. (a)
- p(best parse) is so small as a result of a series of probability multiplications:
  - $\frac{1}{3}$(Root -> S.) * 1(S -> NP VP)* $\frac{1}{2}$(NP -> Det Noun) * $\frac{1}{3}$(Det -> the) * $\frac{1}{6}$(Noun -> president) * 1(VP -> Verb NP.)
  - $\frac{1}{5}$(Verb -> ate) * $\frac{1}{2}$(NP -> Det Noun) * $\frac{1}{3}$(Det -> the) * $\frac{1}{6}$ (Noun -> sandwich)
- It is the same chain of probability multiplication
- Given the sentence, there is only one possible parsing using the grammar file. Thus, P(best parse | sentence) = 1

(b) There are two ways of parsing the sentence using the grammar file: [[[[every sandwich] with [a pickle]] [on [the floor]]] [wanted [a president]].] OR [every sandwich] [with [[a pickle] [on [the floor]]]] [wanted [a president]].]
Each of the two ways of parsing has the same probability of 6.202e-10. So given the sentence, the probability of one of the two ways of parsing being best parse is 50%

(c) cross-entropy = (-log(5.144e-05) - log(1.240e-09))/18 = 2.435 bits
(d) perplexity = 2 ^ (cross-entropy) = 2^ 2.435 = 5.4
(e) The compression program struggles to compress the given corpus because it is not possible to achieve the second sentence "The president ate ." with the given grammar. Any VP expansion must contain both a Verb and NP, but the sentence above contains a VP with only a Verb, meaning that by the given grammar rules, the VP expansion is incomplete. So, p(sentence) = 0, with log(0) undefined, and as a result the cross-entropy achieved by the grammar is also undefined. The parser confirms this, showing p(sentence) = 0 and p(best_parse) = 0.

6. (a) Using the command `python3 randsent.py -g grammar2.gr -n 1000 |perl ./parse -P -g grammar2.gr,` we generate 1000 sentences from `grammar2` and then parse according to the very same grammar. After doing so, we obtain an entropy of 2.169 bits per word.

(b) Using the same command, but changing grammars for sentence generation and for parsing to `grammar3,` we obtain an entropy of 2.483 bits per word. This is higher than the entropy for `grammar2.` We expect this because the modifications we made to `grammar2` extend the grammar so as to allow for generation of sentences with new structures; in this way, `grammar3` is more creative and thus should have higher entropy.

(c) When we attempt to compute the entropy of the original `grammar`, the final output tells us that the entropy is infinite bits. What went wrong here was that the original `grammar` generates sentences that exceed the default `max_expansions` limit of 450 very frequently, due to the reasoning discussed in Question 1 of section 2.1. This results in sentences which contain any number of ellipses at its end. Since these ellipses are not found in the `grammar`, the parser assigns `p(sentence)=0` to such sentences, and since the negative log of 0 approaches infinity, we are left with a model entropy of infinity.

7. We first saved the randomly generated corpus (1000 sentences) from grammar2 into a text file so that we can repeat using the same corpus across the three different grammar files.

```
python randsent.py -g grammar2.gr -n 1000 >
grammar2q3-6.txt
```

Then we parsed the corpus using `grammar`, `grammar2` and `grammar3`:

```
cat .\grammar2q3-6.txt | perl ./parse -P -v -g grammar.g
>>>  cross-entropy = 2.650 bits = -(-34650.498 log-prob. /
13076 words)
 cat .\grammar2q3-6.txt | perl ./parse -P -v -g grammar2.g
>>>  cross-entropy = 2.205 bits = -(-28833.363 log-prob. /
13076 words)
cat .\grammar2q3-6.txt | perl ./parse -P -v -g grammar3.gr
 >>>  cross-entropy = 3.102 bits = -(-40566.708 log-prob.
 / 13076 words)
```

The entropy of `grammar2` is indeed lower than the cross entropies for `grammar` and `grammar3.`

4.1 Questions

1. We chose to implement the phenomena given in (a) and (c).

   (a) "a" and "an"
   To get the grammar to use "a" and "an" appropriately, we first recognize that the distinction between the two determiners comes from the sound of the word that follows: "a" is followed by words that begin with consonant sounds, and "an" is followed by words that begin with vowel sounds. More specifically, the words whose starting sounds we are concerned with are the beginning words of what we have previously referred to as Noun expansions. In other words, we are concerned with `Adj` and terminal `Noun`s, such as "president." So, using the naming convention "C" for "consonant" and "V" for "vowel," we first rename all of our current adjectives into `CAdj` (e.g. "fine") and `VAdj` (e.g. "amazing"). Then, we split our previous `Adj → very Adj` rule into `CAdj → very CAdj` and `CAdj → very VAdj`. From here, we can then split our `Noun → Adj Noun` rule into two distinct rules: `CNoun → CAdj Noun` and `VNoun → VAdj Noun`. However, since we now no longer have a `Noun` symbol, we also replace `Noun` in these two rules with `CNoun` and `VNoun`:

   ```
   1     CNoun     CAdj CNoun
   1     CNoun     CAdj VNoun
   1     VNoun     VAdj CNoun
   1     VNoun     VAdj VNoun
   ```

   Now, we have a grammar which is fully capable of using "a" and "an" appropriately.

   (c) Relative clauses
   To further modify the grammar to be able to handle relative clauses for part (c), we represented a relative clause with three rules as follows:

   ```
   1     RC → that TV NP
   1     RC → that NP TV
   1     RC → that NP NCV that NP TV,
   ```

   with `TV` representing transitive verbs and `NCV` representing Noun Clause Verbs. Now, all that is left is to see that the `RC` must be part of a `NP`: namely, it describes a preceding `NP`. So, we add the rule that `NP → NP RC`. Adding these rules allows us to generate the three example sentences.

5 Extra Credit: Extending Further!

We extended our grammar to be able to describe the following additions:

1) Linking verbs that can be directly followed by adjectives.
   Eg: Sally became perplexed .
   the president seemed fine .
2) Verbs that involve object pronouns (me, you, him, her, them)
   Eg: Sally gave him an apple .
   Sally gave an apple to him .
   the president brought me a desk .
   the president brought a desk to me .
3) Modal verbs that can be directly followed by infinitive, transitive, or intransitive verbs
   Eg: Sally can sigh .
   the president must work on the proposal .
4) Causative verbs that are followed by object pronouns and infinitive verbs
   Eg: Sally helped me work on the proposal .
   the president made me want the apple .