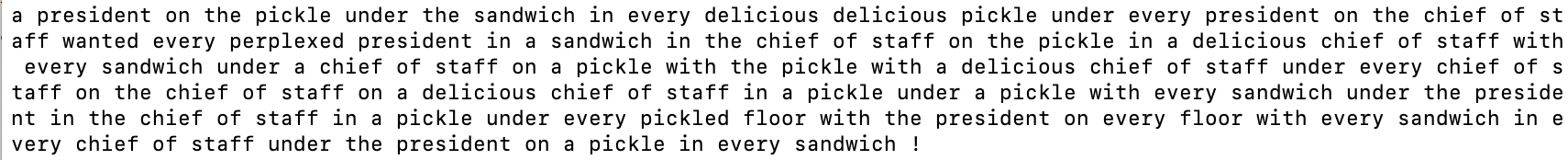
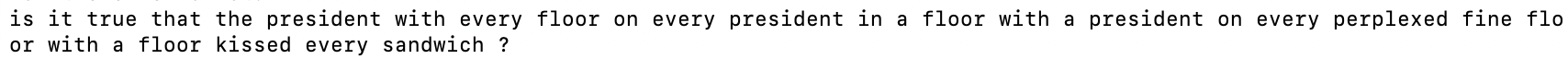
**1.4.1. Provide 10 random sentences generated from your script.**

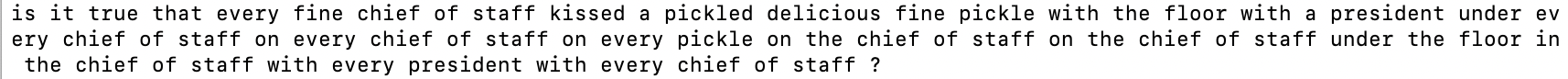


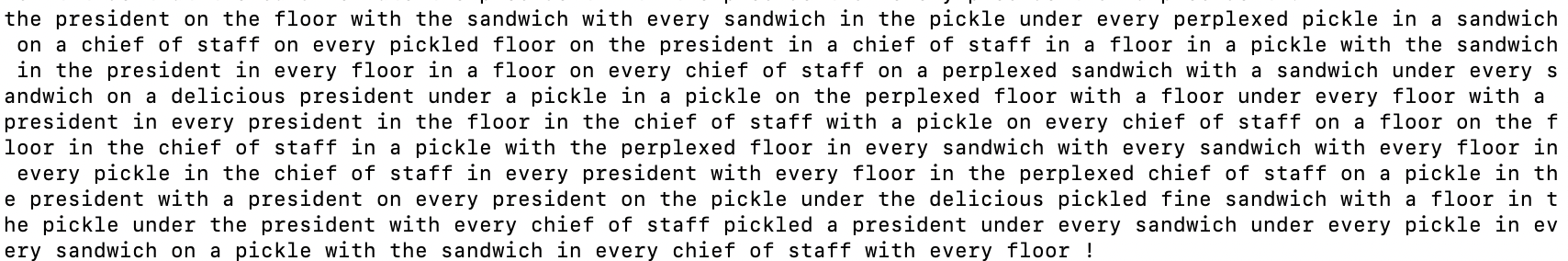


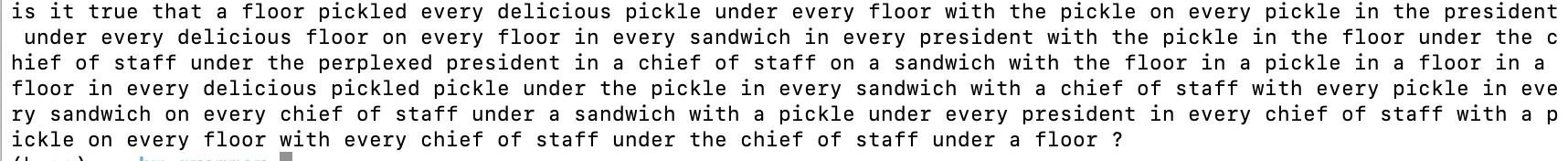




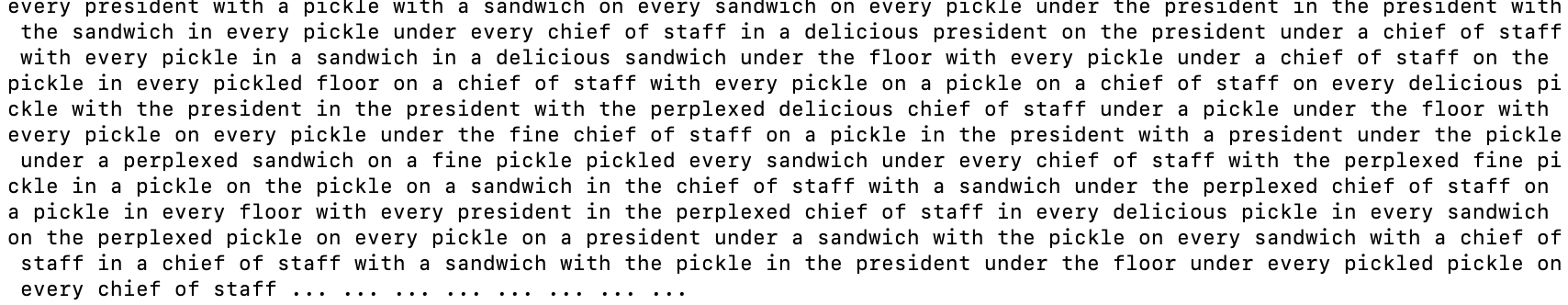












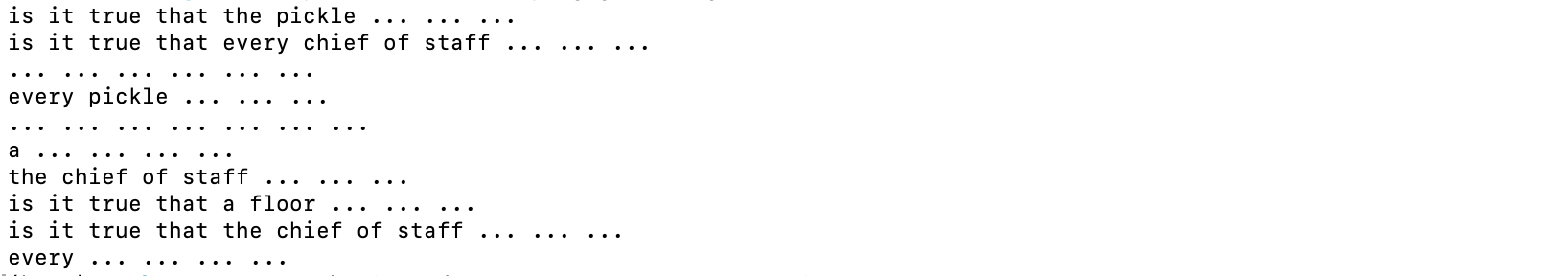
**1.4.2. Provide 2 random sentences generated from your script, using --tree to show their derivations.**

Text, letter

Description automatically generated



1.4.3**. As in the previous question, but with a --max expansions of 5.**



Background pattern

Description automatically generated with low confidence

2.1.1. Why does your program generate so many long sentences? Specifically, what grammar rule (or

rules) is (or are) responsible and why? What is special about it/them?

Because there are some nonterminal symbols cause multi repetition. For example:



2.1.2. The grammar allows multiple adjectives, as in the fine perplexed pickle. Why do your

program’s sentences do this so rarely? (Give a simple mathematical argument.)

Because following the rules below:

Therefore, multiple adjectives are rare





2.1.3. Which numbers must you modify to fix the problems in item 1 and item 2, making the sentences

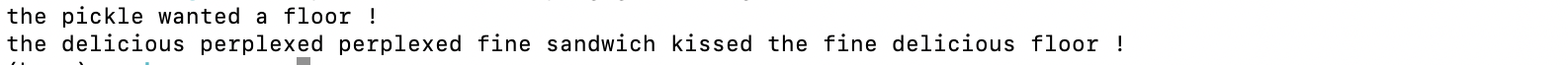
shorter and the adjectives more frequent?

Put these adjustments in a new grammar file named grammar2.gr. Check your answer by running

your generator!



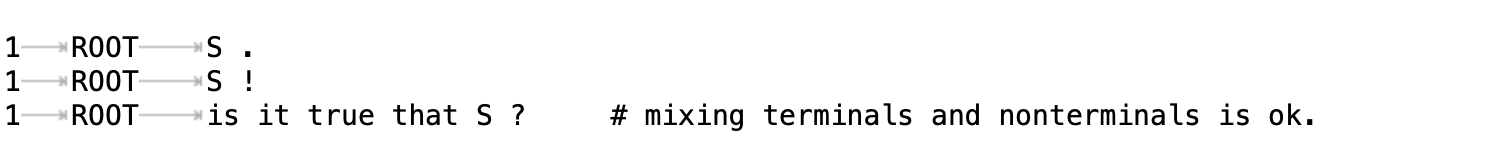




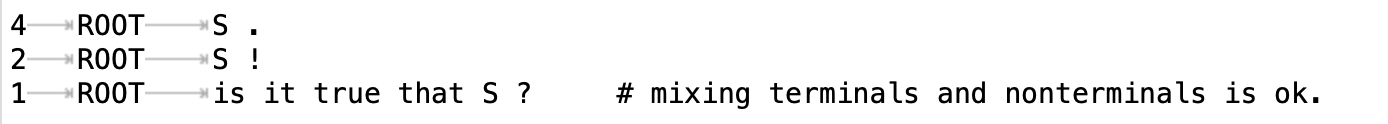
The results are more shorter and have more adjectives

2.1.4. What other numeric adjustments can you make to grammar2.gr in order to favor more natural

sets of sentences? Experiment. Explain the changes.

I’ll change the number of 

to



Since sentences with the declarative structure are much more common in our daily life.

Furthermore, increase the weight for “the”, since phrases like “the president” and “the sandwich” is more common than “a president” or “a sandwich”.

2.1.5. Provide 10 random sentences generated with the grammar2.gr.

Text

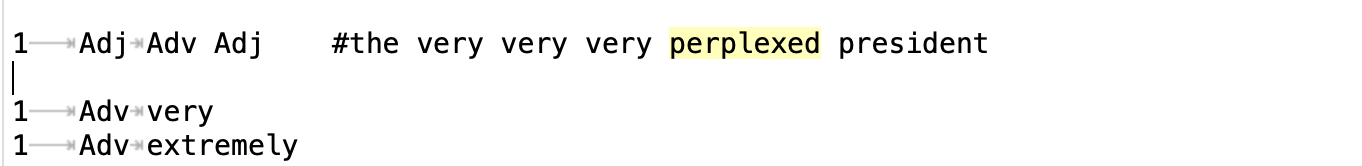
Description automatically generated

2.1.9. Briefly discuss your modifications to the grammar.

* Separate the VP into “IntranVerb” and “tranVerb NP”



* To generate sentences like 4, add “S🡪NP thought that S”
* To generate sentences like 5, add “S🡪it VP that S”
* To generate “very very”, add “Adj🡪Adv Adj” and “Adv🡪extremely” and “Adv🡪very”



* To generate conjunctions, add

Text, letter

Description automatically generated

2.1.10. Provide 10 random sentences generated with grammar3.gr that illustrate your modifications.

Text

Description automatically generated

3.1.1(a) another deviation:

(ROOT (S (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)

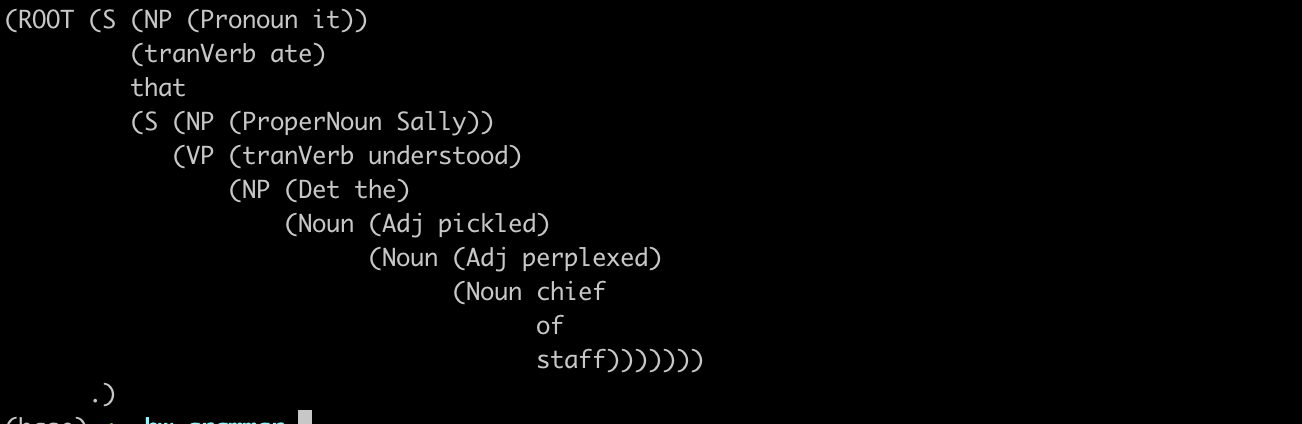
(NP (Det a)

(Noun president)))) .)

3.1.1(b) we must care about that. Because the meaning in different derivations is different. The first derivation means a sandwich on the floor, while the second one means a pickle on the floor

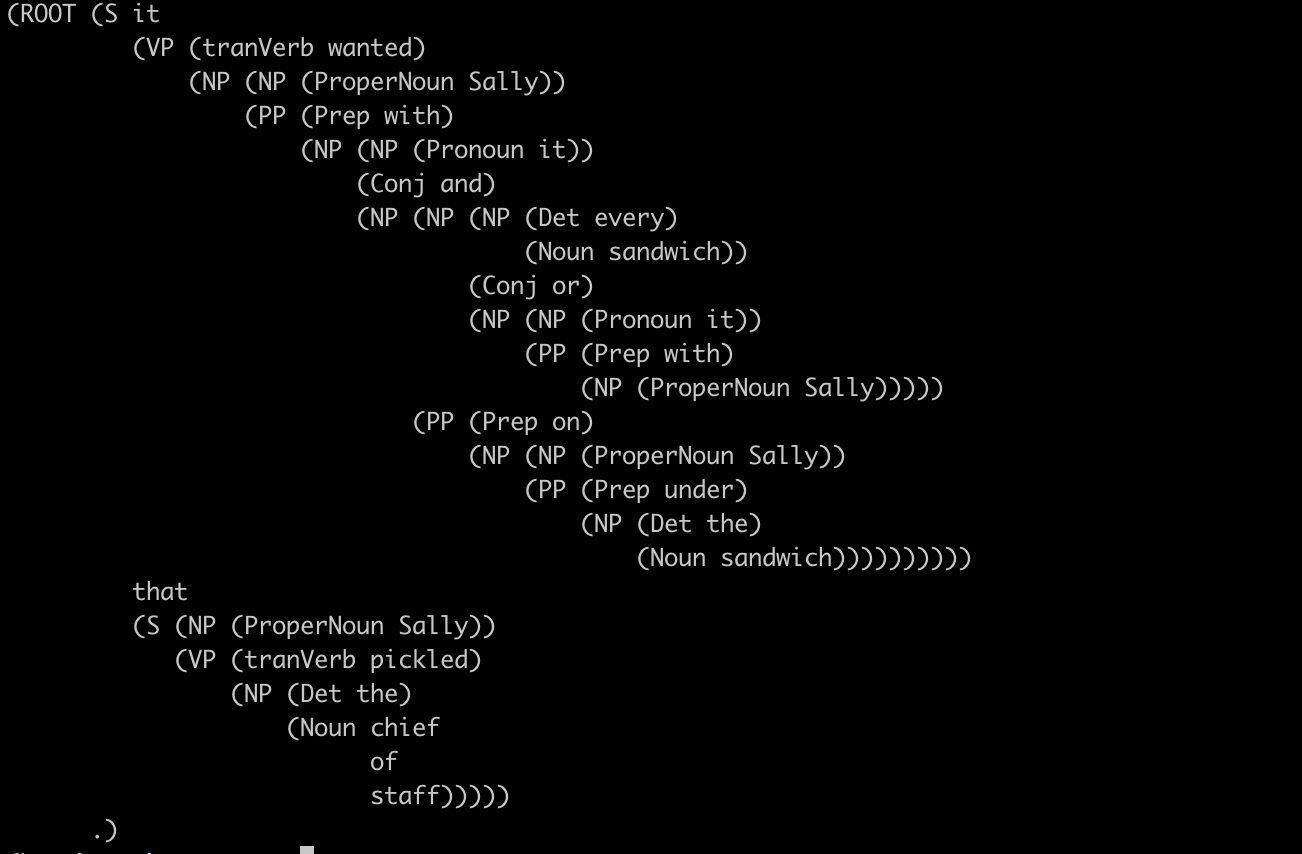
3.3.2.

Example 1: successfully resolve the original derivation



Example 2: it wanted sally with it and every sandwich or it with sally on sally under the sandwich that sally pickled the chief of staff.

In this sentence, there are tons of ambiguities. For example, for sally with it and every sandwich, can be interpreted as “sally and every sandwich” or “it and every sandwich”



3.3.3. there are five ways to analyze the noun phrase below. There are three “prep” in the noun phrase: with on and under. The total ways are 1\*2\*3-1 = 5. Because there is an invalid permutation, which is when on the floor is used to modify every sandwich, under the chief of staff cannot modify a pickle

Text

Description automatically generated

3.3.4.

For grammar.py, the longer the sentence, the more parses. Besides, ambiguities always occur in noun phrase “NP”

For grammar3.py, ambiguities occur in either noun phrases and conjunctions words, such as “and” “or”.

Example 1, in grammar.py, s sentence with 7 “PP” structure. The number of parses is 429

Text

Description automatically generated

Example 2, in grammar3.py



3.3.5 (a)

* because the best\_parse is the only possible way to parse the sentence
* Because the best\_parse is the only parse way, thus, given this sentence, the possibility of best\_parse is equal to 1

3.3.5 (b) There are two possible parse methods for this sentence, and these two ways have the equal possibility. Because the possibility of choosing “NP🡪 Det Noun” and “NP🡪 NP PP” is equal.



3.3.5 (c)

3.3.5 (d)

3.3.5 (e)

p(sentence1) = 5.144e-05

p(sentence 2) = 0

Since the probability of generating sentence 2 is 0, the cross-entropy would be infinity. Therefore, the compression program cannot used to compress the sentence 2, because we would need infinity bits.

3.3.6 (a) We use grammar2 to generate a corpus containing 10000 sentences, with 103587 words. The cross-entropy of grammar2 is 2.203 bits



3.3.6 (b) Similarly, generate 10000 sentences with 107061 words from grammar3, the cross-entropy of grammar3 is 2.500 bits



Grammar3 has higher entropy than grammar2, because we add more sentence structure in grammar3. For example, we split the verb into transitive verb and intransitive verb. We also add a conjunction structure.

3.3.6 (c) When try to compute the entropy of grammar, we got Inf bits. Because grammar tends to generate sentences with “…”. Since “…” didn’t occur in grammar, thus, P(sentence)=0, and cross-entropy = Inf.

3.3.7. Generate a corpus with 10000 sentences from grammar2, then use grammar,grammar2 and grammar3 to compute the cross-entropy.

We find that cross-entropy for grammar2 = 2.199, while cross-entropy for grammar3 is 2.750 bits and for grammar = 2.531, and both are higher than grammar2 itself. Furthermore, we can see that the difference between grammar3 and grammar2 is bigger than grammar and grammar2. We are not surprised by that result, since we changed more on grammar3.













4.1

(b) for the yes-no question, add



Background pattern

Description automatically generated with low confidence

(d) WH-word questions

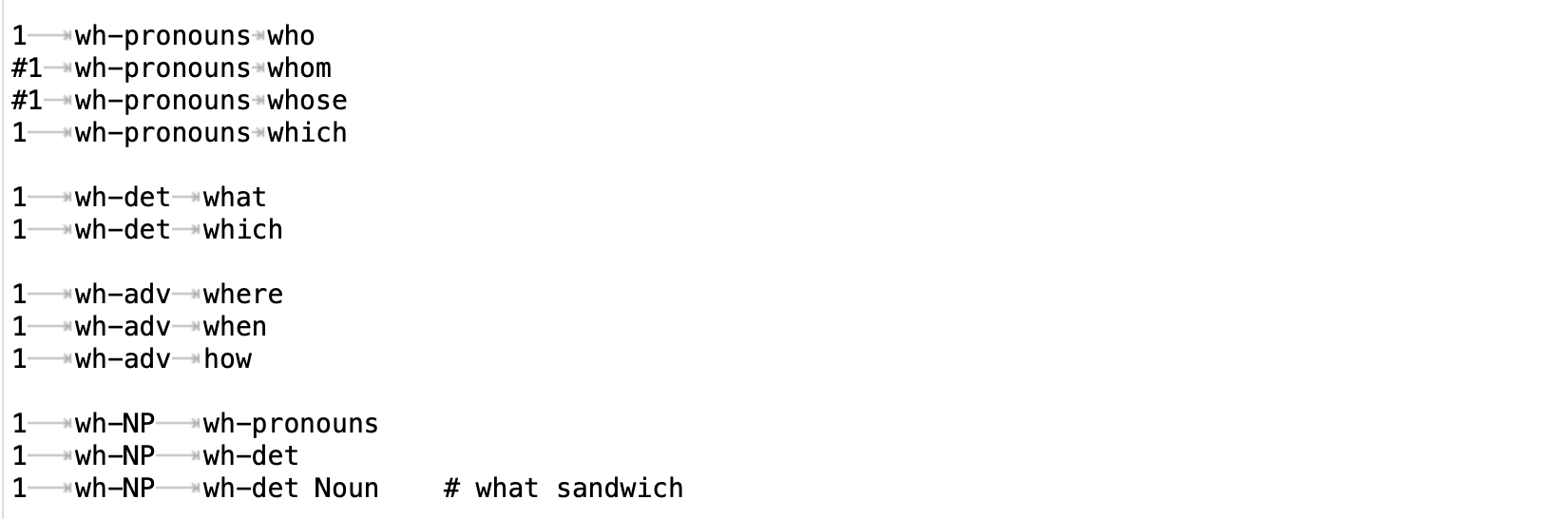
1. We split the wh-words into three categories: wh-pronouns, wh-det and wh-adv

2. Wh-noun can work like the NP in a sentence, and wh-noun can be made by wh- pronouns, wh-det+noun and wh-det itself.

3. there are three wh-sentence structure in grammar4:

* wh-np vp ? this structure is identical to the declarative structure, except that the first noun phrase contains some wh-word
* wh-adv aux s? this structure is to implement sentence like “where did Sally eat the sandwich ?”
* wh-np aux nos? where nos is the sentence without objectives.

Add



and

