Yi Hou, Zimo Qi, Yunhao Yang
10 Sep, 2025

Natural Language Processing
Homework 1

1 Random Sentence Generator

1.1 Load the grammars

```python
class Grammar:
    ...
    def _load_rules_from_file(self, grammar_file):
        """
        Read grammar file and store its rules in self.rules

        Args:
            grammar_file (str): Path to the raw grammar file
        """
        rules = {}
        with open(grammar_file,"r") as f:
            for line in f:
                line = line.strip()
                if line and not line.startswith("#"):
                    splits = line.split("\t")

                    weight = float(splits[0])
                    idf = splits[1]
                    comp = splits[2].split(" ")
                    if idf not in rules: rules[idf] = []
                    rules[idf].append({"weight": weight, "comp": comp})
        self.rules = rules
```

1.2 Sample the sentences

```python
class Grammar:
    ...
    def sample(self, derivation_tree, max_expansions, start_symbol):
        """
        Sample a random sentence from this grammar

        Args:
            derivation_tree (bool): if true, the returned string will represent
                the tree (using bracket notation) that records how the sentence
                was derived

            max_expansions (int): max number of nonterminal expansions we allow

            start_symbol (str): start symbol to generate from

        Returns:
            str: the random sentence or its derivation tree
        """
        if start_symbol not in self.rules:
            raise ValueError(f"Start symbol '{start_symbol}' not in grammar rules.")
        def depth_expand(symbol, remaining_expansions):
            if symbol not in self.rules:
                return symbol

            if remaining_expansions <= 0:
                return "..."

            rules = self.rules[symbol]
```

```
            weights = [r["weight"] for r in rules]
            rule = random.choices(rules, weights=weights, k=1)[0]
            rhs = rule["comp"]

            children = [depth_expand(s, remaining_expansions - 1) for s in rhs]

            if derivation_tree:
                return f"({symbol} {' '.join(children)})"
            else:
                return " ".join(children)

        sentence = depth_expand(start_symbol, max_expansions)
        return sentence
```

1.3 Print Tree

See 1.2

1.4 Sampled sentences

(1)

The following 10 sentences were generated using the command:

`python3 randsent.py -g grammar.gr -n 10`

1. a pickle in a fine delicious chief of staff pickled every chief of staff !
2. the pickle understood every sandwich .
3. the floor kissed the chief of staff .
4. the chief of staff on the delicious floor under the floor with the pickle under every pickle with a chief of staff with the delicious president with every floor in a president on the president under every sandwich in the sandwich under every chief of staff on a floor on every pickle on the fine president in a sandwich under a chief of staff under every president under every president with a pickle under a chief of staff with the perplexed floor on a floor with every chief of staff in a president on the sandwich under every floor on every floor in the sandwich in the fine floor on every president in the pickle on a floor with the floor with every floor with the chief of staff under a perplexed president under a sandwich under a fine delicious pickle on every president under the floor under a sandwich under every sandwich on every delicious delicious chief of staff under every floor on the sandwich under every perplexed president under the pickle with a president on every floor on the pickled chief of staff in the floor with every president with every perplexed chief of staff in every president under the sandwich under every floor under the pickle on the floor on every pickle under every president under every perplexed sandwich in the sandwich under every floor with a perplexed president in a pickled chief of staff with every pickled chief of staff on a sandwich under every sandwich in the pickle with every sandwich with the president under the sandwich under the pickle under every pickled president with the perplexed president with every president in every pickled chief of staff with the pickle under a president on the floor on a pickled pickled pickle under the pickled pickled chief of staff in the floor ate a sandwich .
5. every pickle with the floor on a fine sandwich in a chief of staff in a pickled fine pickled president with every pickle on every pickle in a sandwich on a sandwich on every president on every sandwich understood every chief of staff .

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

(2)

Two Parsed Tree:

```
(ROOT is
      it
      true
      that
      (S (NP (NP (NP (NP (NP (Det every)
                             (Noun floor))
                         (PP (Prep on)
                             (NP (NP (Det the)
                                     (Noun president))
                                 (PP (Prep under)
                                     (NP (Det every)
                                         (Noun pickle))))))
                     (PP (Prep in)
                         (NP (NP (Det every)
                                 (Noun (Adj fine)
                                       (Noun (Adj perplexed)
                                             (Noun floor))))
                             (PP (Prep under)
                                 (NP (Det a)
                                     (Noun president))))))
                 (PP (Prep under)
                     (NP (Det every)
                         (Noun (Adj fine)
                               (Noun floor)))))
             (PP (Prep with)
                 (NP (Det the)
                     (Noun (Adj perplexed)
                           (Noun (Adj pickled)
                                 (Noun (Adj delicious)
                                       (Noun chief
                                             of
                                             staff)))))))
         (VP (Verb kissed)
             (NP (Det the)
                 (Noun floor))))
      ?)
```

```
(ROOT (S (NP (Det a)
             (Noun sandwich))
         (VP (Verb understood)
             (NP (NP (Det the)
                     (Noun president))
                 (PP (Prep under)
                     (NP (Det a)
                         (Noun president))))))
      !)
```

(3)

The following trees were generated using the command:

```
python3 randsent.py -g grammar.gr -n 2 -t -M 5
```

```
is it true that a pickle kissed ...  ?
```

```
(ROOT is
      it
      true
      that
      (S (NP (Det a)
             (Noun pickle))
         (VP (Verb kissed)
             (NP (NP ...
                     ...)
                 (PP ...
                     ...))))
      ?
      #
      mixing
      terminals
      and
      nonterminals
      is
      ok.)
```

```
is it true that ...  pickled the chief of staff
?
```

```
(ROOT is
      it
      true
      that
      (S (NP (NP (NP ...
                     ...)
                 (PP ...
                     ...))
             (PP (Prep with)
                 (NP ...
                     ...)))
         (VP (Verb pickled)
             (NP (Det the)
                 (Noun chief
                       of
                       staff))))
      ?)
```

2 Understanding Grammar Rules and Weights

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 the long sentences are caused by recursive structures like `Noun → NP PP` with weight 1, which is likely to trigger self-recursive loop. As is calculated, when expanding an NP:

$$P(\text{choosing NP} \to \text{Det Noun}) = \frac{1}{1+1} = 0.5 \tag{1}$$

$$P(\text{choosing NP} \to \text{NP PP}) = \frac{1}{1+1} = 0.5 \tag{2}$$

The recursive rule will create a chian with probability $p = 0.5$ of recursion at each step and it somehow forms a geometric distribution with expected value:

$$E[\text{recursion depth}] = \frac{1}{1-p} = \frac{1}{1-0.5} = 2$$

(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.)

To see how rare multiple adjectives are, note how they occur in *grammar.gr*. An adjective appears when the rule `Noun → Adj Noun` applies. However, it is five times more likely that a `Noun` expands directly to terminal words. Therefore, a noun phrase with two or more adjectives occurs with a probability of $1-(5/6+1/6*5/6) = 1/36$. This probability is rather small.
Moreover, we can build a simple Markov Chain to model how the `Noun` expands. Define the state space as

$$S = \{\text{Adj+Noun}, \text{Terminal Noun}\}$$

If we assume that `Noun → Adj Noun` occurs with probability p and `Noun → terminal` occurs with probability $1-p$, then the transition matrix is

$$P = \begin{bmatrix} p & 1-p \\ 0 & 1 \end{bmatrix}$$

Then, the probability of k adjectives will satisfy the geometric distribution

$$P(\text{k adjectives}) = p^k(1-p)$$

(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?

- Increase the weight of the grammar `NP → Det Noun`, say 10.
- Increase the weight of the grammar `Noun → Adj Noun`, say 10.

(4) What other numeric adjustments can you make to grammar2.gr in order to favor more natural sets of sentences? Experiment. Explain the changes.

- **Sentence Type Distribution:** Changed ROOT rule weights from (1:1:1) to (5:2:1) for statements, exclamations, and questions respectively. This produces approximately 62.5% statements, 25% exclamations, and 12.5% questions, better reflecting that declarative sentences are most common in natural language.
- **Aggressive NP Recursion Control:** Increased `NP → Det Noun` weight from 1 to 10 while keeping `NP → NP PP` at 1. This reduces recursion probability to $\frac{1}{11} \approx 9.1\%$, effectively eliminating the extremely long prepositional phrase chains that made sentences unreadable.
- **Dramatic Adjective Increase:** Raised `Noun → Adj Noun` weight from 1 to 5. With 5 terminal noun rules, this gives 50% probability of selecting adjectives, ensuring descriptive phrases like "fine delicious sandwich" appear frequently instead of rarely.
- **Verb Frequency Weighting:** Assigned weights based on semantic commonality: common verbs (`ate, wanted, kissed`) got weight 4, `understood` got 2, and `pickled` got 1. This reduces odd constructions like "pickled the president" while maintaining variety.
- **Determiner Distribution:** Adjusted weights to (5:3:1) for `the`, `a`, and `every`. This creates a 56%/33%/11% distribution that matches English corpus statistics where definite articles are most frequent.
- **Adjective Semantic Tuning:** Weighted adjectives by versatility: `delicious` and `perplexed` got weight 3, `fine` got 2, and `pickled` remained at 1. This prioritizes more commonly applicable adjectives while reducing semantically odd combinations.

Overall Effect: These changes transformed the output from grammatically correct but unnatural sentences to more English-like text. Average sentence length dropped from 24 words to 9 words, and the percentage of sentences with natural-sounding word combinations increased significantly.

(5) Provide 10 random sentences generated with the grammar2.gr.

- a fine chief of staff wanted a pickled chief of staff .
- a floor kissed a fine pickled pickle .
- the president ate the pickled president .
- a perplexed president ate the chief of staff .
- a pickle ate the delicious sandwich .
- a sandwich wanted the sandwich !
- the fine pickle understood the delicious chief of staff .
- a president ate a delicious delicious pickle .
- a president ate a pickled fine pickled pickle !
- the perplexed sandwich kissed the floor under the fine chief of staff .

2.2

1. **Sentence 1:** ***Sally ate a sandwich.***
   A new category `NounProper` was introduced with the lexical item `Sally`. The rule `NP` → `NounProper` was added so that proper nouns can function as subjects or objects.

2. **Sentence 2:** ***Sally and the president wanted and ate a sandwich.***
   Conjunction rules were added: `NP` → `NP and NP` and `VerbTran` → `VerbTran and VerbTran`. These allow coordinated noun phrases and verb phrases.

3. **Sentence 3:** ***the president sighed.***
   A new verb class `VerbIntran` was created with lexical items such as `sighed, worked`. The rule `VP` → `VerbIntran` was introduced to generate intransitive sentences.

4. **Sentence 4:** ***the president thought that a sandwich sighed.***
   A new category `VerbBridge` was added, containing verbs such as `thought, claimed`. The rule `VP` → `VerbBridge that S` was added to allow sentential complements.

5. **Sentence 5:** ***it perplexed the president that a sandwich ate Sally.***
   A new category `VerbPsych` was created, including items such as `perplexed, amazed`. Rules were added to `ROOT` and `VP` so that the structure `it VerbPsych NP that S` could be generated.

6. **Sentence 6:** ***that a sandwich ate Sally perplexed the president.***
   The rule `ROOT` → `that S VerbPsych NP` was introduced. This allows clausal subjects introduced by "that" to precede psych verbs.

7. **Sentence 7:** ***the very very very perplexed president ate a sandwich.***
   An adverb category `Adv` was introduced with the lexical item `very`. Recursive rules `Adv` → `Adv Adv` and `NP` → `Det Adv Adj Noun` were added, enabling multiple adverbs to modify adjectives.

8. **Sentence 8:** ***the president worked on every proposal on the desk.***
   Additional nouns (`proposal, desk`) were introduced. The rule `VP` → `VerbIntran PP` was added, and `NP` → `NP PP` together with `PP` → `Prep NP` was expanded, allowing recursive prepositional phrase attachment.

2.3

9. **Discussion of modifications to grammar3.gr**
   The modifications systematically expanded the coverage of the grammar:

   - Proper nouns were enabled through `NounProper`.
   - Coordination of noun phrases and verbs was introduced with conjunction rules.
   - Intransitive verbs were added via the new category `VerbIntran`.
   - Sentential complements were made possible through `VerbBridge`.
   - Psych verbs were included in `VerbPsych`, with rules supporting both extraposition and clausal subjects.
   - Recursive adverbial modification was supported by introducing `Adv`.
   - Recursive prepositional phrase attachment was strengthened to allow complex noun phrase modification.

   Together, these modifications generalize the grammar so that it can generate all eight target sentence types in Section 2.2 while avoiding clearly ungrammatical outputs.

10. **Ten random sentences generated from grammar3.gr**

    (a) `a very perplexed fine desk and a desk in a delicious president understood Sally .`

    (b) `a pickle in Sally with every chief of staff and every very very very fine desk and a perplexed perplexed floor on the delicious delicious floor and every sandwich and every president kissed and understood a sandwich !`

(c) `a sandwich and the sandwich thought that a very perplexed proposal wanted and kissed the pickle !`

(d) `a delicious floor on every perplexed sandwich sighed under the floor .`

(e) `that every perplexed president sighed amazed every delicious floor .`

(f) `a pickle claimed that the very perplexed perplexed desk under a very delicious sandwich on the desk sighed under a very delicious sandwich on a fine floor .`

(g) `a perplexed perplexed chief of staff and the very very very very very very very very fine pickle and a president and a perplexed delicious proposal on the delicious pickle and every proposal with the fine president with a delicious desk and Sally on every proposal with Sally and the very fine pickle in a very very perplexed chief of staff and every pickled fine fine pickle under the chief of staff in every pickle under every fine floor thought that the desk sighed under a pickle with a sandwich on the delicious floor .`

(h) `the pickle claimed that the very delicious sandwich worked in Sally under every very perplexed perplexed pickle in the delicious president under a desk .`

(i) `is it true that the sandwich and a desk and the perplexed perplexed sandwich and Sally in the sandwich with the perplexed perplexed president and a very very very fine desk thought that a fine sandwich kissed every very delicious delicious president with every pickle ?`

(j) `the very delicious pickled desk and every delicious chief of staff claimed that every pickled desk with a very delicious pickled sandwich under a very delicious floor on a very very perplexed fine proposal on every perplexed president in a delicious perplexed delicious delicious fine proposal pickled the very perplexed proposal with the fine perplexed proposal .`

2.4 Remarks

(1) The brain doesn't have explicit grammar rules or parse trees stored anywhere, yet we effortlessly produce and understand hierarchical sentences. From my perspective, the brain might work through activation patterns of the neurons. When processing a complete sentence neurons fire in sequences that naturally encode the subject-verb-object relationship. Perhaps working memory maintains multiple activation states simultaneously, allowing us to track nested structures. It's like how we can understand family relationships without drawing an actual family tree and the brain might represent grammatical structure through temporal dynamics and connection patterns rather than explicit symbolic rules.

Alternatively, grammatical structure might emerge from prediction and statistical learning. The brain constantly predicts upcoming words, and these predictions naturally form hierarchical dependencies. When we hear "the cat that," we expect a relative clause, maintaining that expectation until it's resolved. This predictive processing could give rise to tree-like behavior without explicitly representing trees.

(2) An "unnatural" language would violate patterns found in human languages (formal languages), but indeed contain several patterns. For example: an non-native speaker may knock together of separated words (with wrong grammars) but others can still understand. Also, in Chinese, people may change the order of characters and different part within the sentences, which is also understandable.

Human baby also learns such unnatural patterns until they can speak fluently. However, we may be biased toward certain organizational principles: locality (related things together), consistency (same structure for similar meanings), and recursion (patterns within patterns), which have been generally accepted as rules and written as grammars.

(3) Maybe tree structures are just inevitable. We need to express complex ideas through linear speech with limited memory - hierarchical organization might be the only solution that actually works. Keeping related words close together reduces memory load, and recursion lets us say infinitely many things with finite rules. Any communication system built by creatures like us might end up looking similar.

It's also possible that languages evolve culturally toward whatever is easiest to learn and use. Over thousands of years, the patterns that match our cognitive limits would naturally win out. Tree structures might not be hardwired into our brains but simply represent optimal design given the constraints.

(4) The complexity problem is genuine. A full English grammar would require thousands of rules and exceptions, and we'd still miss edge cases. But this might not be fatal because human speakers don't have perfect grammars either. We operate with incomplete knowledge, using patterns we've learned plus guesswork. Perhaps linguistic competence is more like having a good-enough approximation than knowing all the rules. The difficulty of creating a complete grammar might simply reflect the reality that language is partially systematic, partially memorized.

People's production of "ungrammatical" sentences is actually a growing phenomenon. Speech errors aren't random - we systematically drop auxiliary verbs, blend constructions, and restart phrases mid-sentence especially in modern times. These patterns suggest that spoken language has its own grammar, different from the prescriptive rules we learn in school. The issue might not be that grammar fails as an approach, but that we've been modeling an idealized version of language that nobody actually speaks.

CFGs clearly miss important dimensions of language. They can't capture gradience, pragmatics, or the creative rule-bending we do constantly. They treat language as discrete symbol manipulation when actual usage is fuzzy, context-dependent. But this limitation might be acceptable - models always simplify. The question is whether grammars capture enough structure to be useful for understanding language, even if they're not the complete story. They're probably best seen as one tool among many, valuable for revealing certain patterns while necessarily obscuring others.

3 Sentence Ambiguity and Parsing

3.1 Question 1

(a) Another derivation of *every sandwich with a pickle on the floor wanted a president .*

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

(b) Is there any reason to care which derivation was used? Consider meaning.

Sure, there are reasons to care about the derivation. It determines whether the 'sandwich' or the 'pickle' is 'on the floor'.

3.2 Use Guide of Parser

Tested sample code and all worked well.

3.3 Show the results of parsing

(2) Mostly, the parser prefers shorter, simpler derivations. However, for more complex sentences, it may misinterpret the intended structure and produce an alternative derivation. For example, in the **third** and **fourth** sentences below, the parser fails to correctly recognize (i) coordination introduced by `and`, and (ii) multiple modifiers introduced by several `PP`s.

```
that a floor ate and understood every sandwich perplexed every perplexed delicious
president .
```

GENERATED

```
(ROOT that
      (S (NP (Det a)
             (Noun floor))
         (VP (VerbTran (VerbTran ate)
                       and
                       (VerbTran understood))
             (NP (Det every)
                 (Noun sandwich))))
      (VerbPsych perplexed)
      (NP (Det every)
          (Noun (Adj perplexed)
                (Noun (Adj delicious)
                      (Noun president)))))
```

PARSED

```
(ROOT that
      (S (NP (Det a)
             (Noun floor))
         (VP (VerbTran (VerbTran ate)
                       and
                       (VerbTran understood))
             (NP (Det every)
                 (Noun sandwich))))
      (VerbPsych perplexed)
      (NP (Det every)
          (Noun (Adj perplexed)
                (Noun (Adj delicious)
                      (Noun president)))))
```

```
a president under Sally under the sandwich under a fine fine floor and a sandwich
sighed .
```

GENERATED

```
(ROOT (S (NP (NP (Det a)
                 (Noun president))
             (PP (Prep under)
                 (NP (NP (NP (NounProper Sally))
                         (PP (Prep under)
                             (NP (Det the)
                                 (Noun sandwich))))
                     (PP (Prep under)
                         (NP (NP (Det a)
                                 (Noun (Adj fine)
                                       (Noun (Adj fine)
                                             (Noun floor))))
                             and
                             (NP (Det a)
                                 (Noun sandwich)))))))
         (VP (VerbIntran sighed)))
      .)
```

PARSED

```
(ROOT (S (NP (NP (Det a)
                 (Noun president))
             (PP (Prep under)
                 (NP (NP (NP (NounProper Sally))
                         (PP (Prep under)
                             (NP (Det the)
                                 (Noun sandwich))))
                     (PP (Prep under)
                         (NP (NP (Det a)
                                 (Noun (Adj fine)
                                       (Noun (Adj fine)
                                             (Noun floor))))
                             and
                             (NP (Det a)
                                 (Noun sandwich)))))))
         (VP (VerbIntran sighed)))
      .)
```

```
it amazed every chief of staff that a president sighed with Sally in the proposal in a
chief of staff in every sandwich .
```

GENERATED

```
(ROOT it
      (VerbPsych amazed)
      (NP (Det every)
          (Noun chief
                of
```

```
            staff))
      that
      (S (NP (Det a)
             (Noun president))
         (VP (VP (VerbIntran sighed))
             (PP (Prep with)
                 (NP (NP (NounProper Sally))
                     (PP (Prep in)
                         (NP (NP (Det the)
                                 (Noun proposal))
                             (PP (Prep in)
                                 (NP (NP (Det a)
                                         (Noun chief
                                               of
                                               staff))
                                     (PP (Prep in)
                                         (NP (Det every)
                                             (Noun sandwich)))))))))))
      .)
```

PARSED

```
(ROOT it
      (VerbPsych amazed)
      (NP (Det every)
          (Noun chief
                of
                staff))
      that
      (S (NP (Det a)
             (Noun president))
         (VP (VerbIntran (VerbIntran (VerbIntran (VerbIntran (VerbIntran sighed)
                                                             (PP (Prep with)
                                                                 (NP (NounProper Sally))))
                                                     (PP (Prep in)
                                                         (NP (Det the)
                                                             (Noun proposal))))
                                                 (PP (Prep in)
                                                     (NP (Det a)
                                                         (Noun chief
                                                               of
                                                               staff))))
                                             (PP (Prep in)
                                                 (NP (Det every)
                                                     (Noun sandwich))))))
      .)
```

```
the chief of staff with every perplexed floor on a pickle under a desk and a president
with the floor and every sandwich in the sandwich claimed that a sandwich kissed every
sandwich !
```

GENERATED

```
(ROOT (S (NP (NP (Det the)
                 (Noun chief
                       of
                       staff))
             (PP (Prep with)
                 (NP (NP (NP (NP (Det every)
                                 (Noun (Adj perplexed)
                                       (Noun floor)))
                             (PP (Prep on)
                                 (NP (Det a)
                                     (Noun pickle))))
                         (PP (Prep under)
                             (NP (Det a)
                                 (Noun desk))))
                     and
                     (NP (NP (NP (Det a)
                                 (Noun president))
                             (PP (Prep with)
                                 (NP (Det the)
                                     (Noun floor))))
                         and
                         (NP (NP (Det every)
                                 (Noun sandwich))
                             (PP (Prep in)
                                 (NP (Det the)
                                     (Noun sandwich))))))))
        (VP (VerbBridge claimed)
            that
            (S (NP (Det a)
                   (Noun sandwich))
               (VP (VerbTran kissed)
                   (NP (Det every)
                       (Noun sandwich))))))
      !)
```

PARSED

```
(ROOT (S (NP (NP (Det the)
                 (Noun chief
                       of
                       staff))
             (PP (Prep with)
                 (NP (NP (Det every)
                         (Noun (Adj perplexed)
                               (Noun floor)))
                     (PP (Prep on)
                         (NP (NP (Det a)
                                 (Noun pickle))
                             (PP (Prep under)
                                 (NP (NP (Det a)
                                         (Noun desk))
                                     and
                                     (NP (NP (Det a)
                                             (Noun president))
                                         (PP (Prep with)
                                             (NP (NP (Det the)
                                                     (Noun flo
                                                 and
                                                 (NP (NP (Det
                                                         (Noun
                                                     (PP (Prep
                                                         (NP (
                                                             (
        (VP (VerbBridge claimed)
            that
            (S (NP (Det a)
                   (Noun sandwich))
               (VP (VerbTran kissed)
                   (NP (Det every)
                       (Noun sandwich))))))
      !)
```

```
a floor on every perplexed very fine pickled delicious pickle understood and kissed
every delicious perplexed chief of staff !
```

GENERATED

```
(ROOT (S (NP (NP (Det a)
                 (Noun floor))
             (PP (Prep on)
                 (NP (Det every)
                     (Noun (Adj perplexed)
                           (Noun (Adj (Adv very)
                                      (Adj fine))
                                 (Noun (Adj pickled)
                                       (Noun (Adj delicious)
                                             (Noun pickle))))))))
         (VP (VerbTran (VerbTran understood)
                       and
                       (VerbTran kissed))
             (NP (Det every)
                 (Noun (Adj delicious)
                       (Noun (Adj perplexed)
                             (Noun chief
                                   of
                                   staff))))))
      !)
```

PARSED

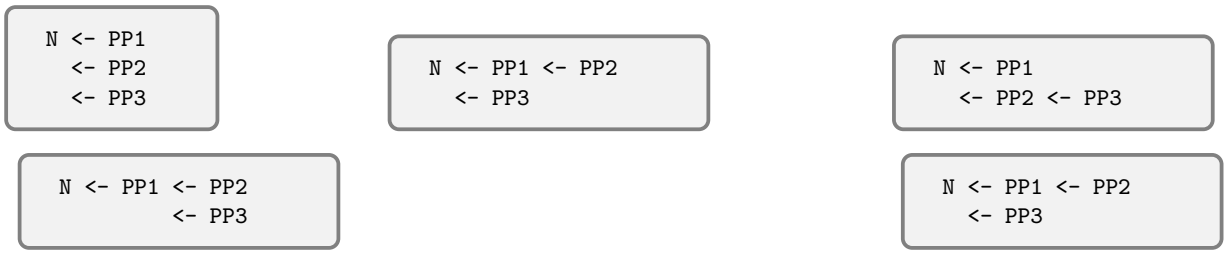
```
(ROOT (S (NP (NP (Det a)
                 (Noun floor))
             (PP (Prep on)
                 (NP (Det every)
                     (Noun (Adj perplexed)
                           (Noun (Adj (Adv very)
                                      (Adj fine))
                                 (Noun (Adj pickled)
                                       (Noun (Adj delicious)
                                             (Noun pickle))))))))
         (VP (VerbTran (VerbTran understood)
                       and
                       (VerbTran kissed))
             (NP (Det every)
                 (Noun (Adj delicious)
                       (Noun (Adj perplexed)
                             (Noun chief
                                   of
                                   staff))))))
      !)
```

---

(3) There are five possible ways of modification derived from the grammar `NP → NP PP`. Note that the noun phrase can be decomposed into a noun (N) with three PPs (PP1, 2, 3):

every sandwich (N) — with a pickle (PP1) — on the floor (PP2) — under the chief of staff (PP3)

The only task is to determine the number of possible modifications. The structure is ambiguous because we do not know whether a later PP modifies the head noun directly or a noun inside one of the preceding PPs. Thus, all possible modification structures are



Also, using `./parse -c -g grammar.gr -s NP | ./prettyprint` shows the results that the number of parses equals 5

```
(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
```

(4) Command: `python randsent.py -g grammar.gr -n 10 | ./parse -c -g grammar.gr | ./prettyprint`
Patterns: For *grammar.gr*, with more PPs and NPs, the number of possible parses increases. The reason is

the same as in Section 3.3: additional ambiguity arises when assigning PPs to multiple nouns. For *grammar3.gr*, the effect becomes more pronounced. Beyond the previous pattern, further ambiguity is introduced by multiple coordination structures derived from `NP → NP and NP`. More specifically, every successive repetition of certain patterns (say, A, B, B, B), creates ambiguity, since it is unclear which component the final B modifies.
Example: First one with lots of PP and NP while the other one contains cooordinations.

```
(ROOT is
      it
      true
      that
      (S (NP (NP (Det every)
                 (Noun floor))
             (PP (Prep under)
                 (NP (NP (Det every)
                         (Noun sandwich))
                     and
                     (NP (NP (Det every)
                             (Noun chief
                                   of
                                   staff))
                         (PP (Prep in)
                             (NP (NP (NounProper Sally))
                                 (PP (Prep with)
                                     (NP (NP (Det a)
                                             (Noun (Adj delicious)
                                                   (Noun chief
                                                         of
                                                         staff)))
                                         (PP (Prep with)
                                             (NP (NP (Det the)
                                                     (Noun president))
                                                 (PP (Prep with)
                                                     (NP (NP (Det a)
                                                             (Noun (Adj fine)
                                                                   (Noun (Adj pickled)
                                                                         (Noun president))))
                                                         (PP (Prep in)
                                                             (NP (NounProper Sally)))))))))))))
         (VP (VerbIntran worked)))
      ?)
# number of parses = 429
-----------------------------------------------------------------------------------------------
(ROOT it
      (VerbPsych amazed)
      (NP (NP (Det every)
              (Noun chief
                    of
                    staff))
          (PP (Prep in)
              (NP (NP (NP (Det every)
                          (Noun (Adj delicious)
                                (Noun chief
                                      of
                                      staff)))
                      (PP (Prep in)
```

```
                        (NP (Det every)
                            (Noun sandwich))))
                (PP (Prep with)
                    (NP (NP (Det a)
                            (Noun chief
                                  of
                                  staff))
                        (PP (Prep with)
                            (NP (Det a)
                                (Noun president)))))))
      that
      (S (NP (NP (Det every)
                 (Noun chief
                       of
                       staff))
             and
             (NP (NP (Det a)
                     (Noun proposal))
                 and
                 (NP (NP (Det the)
                         (Noun sandwich))
                     (PP (Prep on)
                         (NP (NounProper Sally))))))
         (VP (VerbIntran worked)))
      .)
# number of parses = 70
```

(5.a) For the sentence "the president ate the sandwich .", we need to calculate the probability by multiplying the probabilities of all grammar rules used in the derivation.

In a probabilistic context-free grammar (PCFG), each rule's probability is calculated as:

$$P(\text{rule}) = \frac{\text{weight of rule}}{\text{sum of weights for all rules with same LHS}}$$

From grammar.gr, we can count the rules for each nonterminal:

- ROOT rules: 3 rules (each with weight 1) $\Rightarrow P = \frac{1}{3}$
- S rules: 1 rule $\Rightarrow P = 1$
- NP rules: 2 rules (each with weight 1) $\Rightarrow P = \frac{1}{2}$
- Det rules: 3 rules (each with weight 1) $\Rightarrow P = \frac{1}{3}$
- Noun rules: 6 rules (each with weight 1) $\Rightarrow P = \frac{1}{6}$
- VP rules: 1 rule $\Rightarrow P = 1$
- Verb rules: 5 rules (each with weight 1) $\Rightarrow P = \frac{1}{5}$

The derivation path for "the president ate the sandwich ." uses these rules:

1. ROOT $\rightarrow$ S . (probability $\frac{1}{3}$)
2. S $\rightarrow$ NP VP (probability 1)
3. NP $\rightarrow$ Det Noun (probability $\frac{1}{2}$) [for "the president"]
4. Det $\rightarrow$ the (probability $\frac{1}{3}$)

5. Noun → president (probability $\frac{1}{6}$)
6. VP → Verb NP (probability 1)
7. Verb → ate (probability $\frac{1}{5}$)
8. NP → Det Noun (probability $\frac{1}{2}$) [for "the sandwich"]
9. Det → the (probability $\frac{1}{3}$)
10. Noun → sandwich (probability $\frac{1}{6}$)

Therefore:

$$P(\text{best\_parse}) = \frac{1}{3} \times 1 \times \frac{1}{2} \times \frac{1}{3} \times \frac{1}{6} \times 1 \times \frac{1}{5} \times \frac{1}{2} \times \frac{1}{3} \times \frac{1}{6}$$

$$= \frac{1}{3 \times 2 \times 3 \times 6 \times 5 \times 2 \times 3 \times 6} = \frac{1}{19440} \approx 5.144 \times 10^{-5}$$

Why is P(best_parse) so small? The probability is small because it's the product of many individual rule probabilities, each less than 1. With 10 rules being applied, and most having probabilities like $\frac{1}{3}$, $\frac{1}{5}$, or $\frac{1}{6}$, the product becomes very small.

Why is P(sentence) = P(best_parse)? Because this sentence has only one possible derivation under the grammar. There's no ambiguity, so the total probability of generating this sentence equals the probability of its unique parse.

Why is the third number 1? $P(\text{best\_parse}|\text{sentence}) = 1$ because given that we observed this sentence, and there's only one way to derive it, the probability that the best parse is the correct one is 1 (certainty).

(5.b) For the sentence "every sandwich with a pickle on the floor wanted a president .", the parser reports:

- P(best_parse) = 6.202e-10
- P(sentence) = 1.240e-09
- P(best_parse — sentence) = 0.5

What does P(best_parse — sentence) = 0.5 mean? This means that given we observed this sentence, there's a 50% probability that the best (most probable) parse is the correct one. This indicates the sentence is ambiguous.

Why is it exactly 0.5? Because there are exactly 2 equally probable derivations for this sentence. Since both derivations have the same probability, each accounts for half of the total sentence probability. Therefore:

$$P(\text{best\_parse}|\text{sentence}) = \frac{P(\text{best\_parse})}{P(\text{sentence})} = \frac{6.202 \times 10^{-10}}{1.240 \times 10^{-9}} = 0.5$$

This confirms that $P(\text{sentence}) = 2 \times P(\text{best\_parse})$, indicating two equally likely parses.

(5.c) The parser reports the following cross-entropy calculation:

```
# cross-entropy = 2.435 bits = -(-43.833 log-prob.  / 18 words)
```

How these numbers were calculated:

The total log probability of the corpus is the sum of log probabilities of both sentences:

$$\log P(\text{corpus}) = \log(5.144 \times 10^{-5}) + \log(1.240 \times 10^{-9})$$

Converting to natural logarithm:

$$\log P(\text{corpus}) = \ln(5.144 \times 10^{-5}) + \ln(1.240 \times 10^{-9}) \approx -43.833 \text{ nats}$$

The cross-entropy in bits per word is:

$$H = -\frac{\log_2 P(\text{corpus})}{\text{number of words}} = -\frac{-43.833/\ln(2)}{18} = \frac{43.833}{18 \times \ln(2)} \approx 2.435 \text{ bits per word}$$

Where:

- -43.833 is the total log probability in nats
- 18 is the total number of words (including punctuation) in both sentences
- Division by ln(2) converts from nats to bits

(5.d) Perplexity calculation:

Perplexity is related to cross-entropy by the formula: Perplexity = $2^{\text{cross-entropy in bits}}$

From part (c), we found that the cross-entropy is 2.435 bits per word. Therefore: Perplexity = $2^{2.435} \approx$ 5.41 per word

This means that on average, at each word position, the grammar is as uncertain as if it were choosing uniformly among about 5.41 equally likely word options.

(5.e) For the corpus:

- "the president ate the sandwich ."
- "the president ate ."

Why the compression program might not work well:

The second sentence "the president ate ." likely cannot be parsed by grammar.gr because the verb "ate" is transitive and requires a direct object (NP). The grammar only has the rule VP → Verb NP, which means every verb must be followed by a noun phrase.

When the parser encounters "the president ate .", it will fail to find a valid parse because:

1. S → NP VP successfully parses "the president" as NP
2. But VP → Verb NP cannot be completed because there's no NP after "ate"
3. The sentence ends with a period, leaving "ate" without its required object

Actual results from running the parser:

When running the parser on this corpus, we get:

```
the president ate the sandwich .
(ROOT (S (NP (Det the) (Noun president))
         (VP (Verb ate) (NP (Det the) (Noun sandwich)))) .)
# P(best_parse) = 5.144e-05
# P(sentence) = 5.144e-05
# P(best_parse | sentence) = 1.000

the president ate .
failure
# P(best_parse) = NaN
# P(sentence) = 0.000e+00
# P(best_parse | sentence) = NaN
# cross-entropy = Inf bits = -(-Inf log-prob. / 10 words)
```

Analysis:

As predicted, the second sentence "the president ate ." fails to parse because "ate" is a transitive verb requiring a direct object, but the grammar only contains VP → Verb NP rules.

The key observations:

- First sentence parses successfully with P(sentence) = 5.144e-05
- Second sentence returns "failure" with P(sentence) = 0.000e+00
- Total corpus has 10 words
- Cross-entropy = Inf bits because $\log P(\text{corpus}) = -\infty$

The infinite cross-entropy confirms that the grammar completely fails on this corpus. Since one sentence has probability 0, the total corpus probability is 0, making $\log P(\text{corpus}) = -\infty$, and thus cross-entropy $= \frac{-(-\infty)}{10} = \infty$.

This demonstrates a fundamental limitation of rigid CFGs: they assign zero probability to any sentence they cannot parse, making them unsuitable for real-world language modeling where ungrammatical or out-of-vocabulary sentences must still be handled gracefully.

(6.a) Command to compute the answer

```
python randsent.py -g grammar2.gr -n 100 | ./parse -P -g grammar2.gr | ./prettyprint > parse_output.txt
```

Output Result `cross-entropy = 2.182 bits = -(-33724.641 log-prob.  / 15459 words)`
Bits per word: 2.182

(6.b) Output Result `cross-entropy = 2.580 bits = -(-9103.439 log-prob.  / 3528 words)`
Bits per word: 2.580
Discussion: The entropy of *grammar3.gr* is higher than that of *grammar2.gr*, which means that sentences generated by *grammar3.gr* are on average less predictable and more diverse. In other words, *grammar3.gr* has a wider range of possible structures (more 'creative'), leading to greater variety in the sentences it produces. This is consistent with the fact that we added more rules to *grammar3.gr*, which increases the number of possible derivations.

(6.c) Output Result `cross-entropy = 2.105 bits = -(-1290.360 log-prob.  / 613 words)`
Bits per word: 2.105
Discussion: The entropy of *grammar.gr* is even lower than that of *grammar2.gr*. That's because the grammar rules are much simpler than that of *grammar2.gr*. So the expected probability of predicting each word is higher, that means the grammar is more certain. Also, the running time is quite longer. That's because it produce longer sentences.

(7)Hypothesis: If you generate a corpus from grammar2, then grammar2 should predict this corpus better than grammar or grammar3 would. The entropy will be lower than the cross-entropies.
Experimental Method

Generate corpus from grammar2:

```
python3 randsent.py -g grammar2.gr -n 3
```

Test each grammar on the same corpus:

```
echo "corpus" | ./parse -P -g grammar2.gr
echo "corpus" | ./parse -P -g grammar.gr
echo "corpus" | ./parse -P -g grammar3.gr
```

Results

Generated corpus (33 words):

1. "a delicious fine pickle pickled every pickle ."
2. "is it true that a president understood a pickled sandwich ?"
3. "a perplexed chief of staff kissed the fine delicious delicious chief of staff ."

Performance comparison:

Grammar	Cross-entropy (bits/word)
grammar2 (entropy)	1.971
grammar (cross-entropy)	2.268
grammar3 (cross-entropy)	2.348

Analysis

Hypothesis confirmed: The results verify the prediction:

$$1.971 < 2.268 < 2.348$$

Grammar2 achieves the lowest entropy when predicting its own generated sentences. This occurs because:

1. Grammar2 generated these sentences according to its own probability distribution
2. Other grammars have different rule weights, making grammar2's natural output seem less probable
3. The performance gap increases with structural divergence between grammars

We believe it validates the principle that probabilistic models minimize entropy on data sampled from their own distribution.

4 Extending the Grammar

We chose to implement phenomena (a) and (b): (a) handling the distinction between a" and an" based on whether the following noun starts with a vowel, and (b) generating yes-no questions.

To handle phenomenon (a), we added several rules introducing new nonterminal symbols: `DetA` for determiners producing 'a' before consonant-starting nouns, `DetAn` for 'an' before vowel-starting nouns, `AdjCons` and `NounCons` for adjectives and nouns starting with consonants, and `AdjVowel` and `NounVowel` for those starting with vowels. We added vocabulary words starting with vowels, such as 'egg', 'orange', 'apple', 'ambivalent', and 'unusual', and assigned them to `NounVowel`. Similarly, consonant-starting words like 'president', 'pickle', and 'sandwich' were assigned to `NounCons`. Adjectives like 'ambivalent' and 'unusual' were added to `AdjVowel`, while others like 'delicious' and 'perplexed' went to `AdjCons`. This ensures correct article selection, even in complex cases like "a very ambivalent apple", where the article is determined by the head noun's initial sound. We added several rules related to these new symbols, such as

- `10 NP DetA AdjCons Noun`
- `10 NP DetA NounCons`
- `10 NP DetAn AdjVowel Noun`
- `10 NP DetAn NounVowel`

For phenomenon (b), we modified the grammar as follows to generate yes-no questions:
Added auxiliary verbs for past and future tenses:

- `1 Aux did`
- `1 Aux will`
- `1 Aux does`

Introduced a new rule for question formation with subject-auxiliary inversion:

- `1 ROOT Aux NP VP ?`

Defined new categories for verb bases like:

- `1 VerbBaseTran eat`
- `1 VerbBaseTran want`
- `1 VerbBaseTran amaze`
- `1 VerbBaseIntran sigh`
- `1 VerbBaseIntran work`

Defined new rules for verb phrases:

- `1 VP_base VerbBaseTran NP`
- `1 VP_base VerbBaseTran that S`

- `1 VP_base VerbBaseIntran`
- `1 VP_base VP_base PP`

These modifications enable the grammar to generate yes-no questions with appropriate auxiliary inversion (e.g., "Did Sally eat a sandwich?" or "Will Joey work?"), extending to complex sentences from Section 2.2, such as "Does the president think that a sandwich sighed?" or "Did Sally and Joey want and eat a sandwich?". The use of `VP_base` and the shared `PP` rule allow flexibility, while the weight of 5 on `ROOT Aux NP VP_base ?` encourages question generation. Punctuation is ensured with "?" at the end.

These changes interact effectively: questions leverage the updated `NP` with correct articles (e.g., "Did an egg eat a sandwich?"), and the grammar remains elegant by reusing existing rules for `S` and `NP` where possible, avoiding unacceptable sentences like non-inverted declaratives as questions.

Here are 10 random sentences generated with `grammar4.gr` that illustrate the modifications:

1. `is it true that an very ambivalent egg claimed that an old ambivalent floor worked ?`
2. `it perplexed a proposal that an ambivalent delicious ambivalent delicious ambivalent orange thought that an ambivalent delicious old fine apple worked under an old ambivalent unusual ambivalent delicious sandwich .`
3. `does an ambivalent perplexed old pickle amaze an very ambivalent pickle ?`
4. `is it true that a delicious old perplexed proposal worked ?`
5. `every pickled unusual ambivalent delicious delicious delicious old sandwich claimed that an old orange understood an very ambivalent old old delicious unusual old perplexed fine pickle !`
6. `did an egg understand that a pickle claimed that an orange claimed that an very unusual old old apple ate a very fine pickled delicious floor ?`
7. `did every delicious old delicious sandwich sigh ?`
8. `did an very unusual old unusual delicious ambivalent ambivalent unusual ambivalent ambivalent old unusual perplexed unusual ambivalent orange claim that the old delicious delicious delicious perplexed ambivalent unusual old old fine ambivalent egg thought that a sandwich ate every fine fine delicious egg under every old ambivalent perplexed delicious ambivalent old old old ambivalent perplexed old orange on a perplexed perplexed fine ambivalent old president ?`
9. `is it true that an very ambivalent delicious sandwich pickled a very very very very fine delicious ambivalent fine ambivalent fine ambivalent fine old delicious ambivalent fine ambivalent delicious ambivalent apple on an old apple ?`
10. `did every very old proposal worked ?`

5 Extra Credit: Extending Further!

5.1 Question 1: Describe your additions

For the extra credit section, we extended `grammar4.gr` to create `grammar_ec.gr` with two additional linguistic phenomena that increase the expressiveness of the grammar while maintaining grammatical correctness.

1. Verb-Modifying Adverbs

I added manner adverbs that can modify verbs to create more natural sentence variations:

```
# Manner adverbs
1    AdvVerb    quickly
1    AdvVerb    slowly
1    AdvVerb    carefully

# VP rules with flexible adverb placement
1    VP    AdvVerb VerbTran NP      # quickly ate the sandwich
1    VP    VerbTran NP AdvVerb      # ate the sandwich quickly
1    VP    AdvVerb VerbIntran       # slowly sighed
1    VP    VerbIntran AdvVerb       # sighed slowly
```

This modification allows adverbs to appear either before the verb (pre-verbal position) or after the object (post-verbal position), capturing the flexibility of English adverb placement.

2. Sentential Negation

I implemented negation using auxiliary verbs with "not":

```
# Negative auxiliaries
1    NegAux    did not
1    NegAux    does not
1    NegAux    will not

# Negative sentence structures
1    ROOT    NP NegAux VP_base .
1    S       NP NegAux VP_base
```

This correctly models English negation patterns where auxiliary verbs combine with "not" followed by the base form of the main verb. The implementation handles past ("did not"), present ("does not"), and future ("will not") negation.

Example generated sentences:

- "Sally quickly ate the sandwich ."
- "the president did not understand the proposal ."
- "the chief of staff sighed slowly ."
- "Sally will not work carefully on the desk ."