1 Context Free Grammar exercise

These are outputs to constructing a CFGs following exercise.

1.4.1. 10 random sentences.

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

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


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

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

```
every sandwich wanted the floor with the pickle in every fine floor in
     every floor with a chief of staff !


is it true that the pickle with the president in the floor on every
    pickle with the pickle with a sandwich under a pickle on the chief
     of staff under a delicious president on a floor with the
    president with the perplexed chief of staff under the president in
     every perplexed floor with the sandwich under every president on
    the president with a chief of staff in every chief of staff with
    every pickle in a delicious chief of staff in the president
    understood a president ? # mixing terminals and nonterminals is ok
    .

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

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

```
     sandwich under a pickled pickle under every pickle on a floor
    under every president on every sandwich on every floor on every
    pickled sandwich under the sandwich under the chief of staff on
    every chief of staff with a pickle under the chief of staff with a
     pickle on every pickle with every president in the sandwich in
    every sandwich in the sandwich with the pickle in the president on
     every floor under every fine president on the floor under a
    sandwich in a chief of staff on the chief of staff under the
    pickle with the sandwich on every pickled floor with every floor
    with every sandwich under the sandwich in the pickled chief of
    staff in every president on the pickle on a chief of staff under a
     pickle on every president on every sandwich on every pickle with
    the pickle in the chief of staff under a pickle in the floor under
     the pickle on every floor in every president under every
    perplexed chief of staff in every sandwich under a floor ? #
    mixing terminals and nonterminals is ok.

is it true that every perplexed delicious sandwich pickled a sandwich
    ? # mixing terminals and nonterminals is ok.

a pickle ate a president .
```

1.4.2: 2 random setnences with –tree.

```
python3 randsent.py -g grammar.gr -n 2 --tree
```

```
    (ROOT is
      it
      true
      that
      (S (NP (Det the)
             (Noun president))
         (VP (Verb ate)
             (NP (Det every)
                 (Noun pickle))))
      ?
      #
      mixing
      terminals
      and
      nonterminals
      is
      ok.)

(ROOT (S (NP (NP (Det every)
                 (Noun pickle))
             (PP (Prep on)
                 (NP (Det every)
                     (Noun floor))))
         (VP (Verb ate)
             (NP (Det every)
                 (Noun floor)))))
```

```
      !)
```

1.4.3: 1 and 2 repeated with –max.expansion of 5

```
python3 randsent.py -g grammar.gr -n 10 -M 5
```

```
    every pickle kissed ... ... !

... ... ... ... ... ... ... kissed ... ... ... ... !

is it true that ... ... ... ... ... kissed ... ... ? # mixing
    terminals and nonterminals is ok.

is it true that ... ... ... ... ... wanted ... ... ? # mixing
    terminals and nonterminals is ok.

is it true that ... ... ... ... ... ... ... pickled ... ... ... ... ?
    # mixing terminals and nonterminals is ok.

is it true that a floor kissed ... ... ... ... ? # mixing terminals
    and nonterminals is ok.

... ... ... ... ... understood ... ... ... .

... ... ... ... ... ... ... understood ... ... ... ... .

a ... ... ate ... ... ... ... .

... ... ... ... ... ... ... pickled ... ... ... ... !
```

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

```
(ROOT (S (NP (NP (Det ...)
                 (Noun ...))
             (PP (Prep ...)
                 (NP ...
                     ...)))
         (VP (Verb pickled)
             (NP (Det ...)
                 (Noun ...))))
      !)
(ROOT (S (NP (NP (NP ...
                     ...)
                 (PP ...
                     ...))
             (PP (Prep ...)
                 (NP ...
                     ...)))
         (VP (Verb kissed)
             (NP (Det ...)
                 (Noun ...
```

```
                        ...))))
      .)
```

2 Grammar rules and weights

2.1.1 One observes that the program generate long sentences. This is likely because of the *recursive* nature of the rules. For example the rule

```
    NP → Det Nomininal
    Nominal → Noun
    Nominal → Nominal Noun
```

in our case,

```
1   NP   NP PP
```

2.1.2. The grammar rule

```
1   Noun   Adjective Noun
```

allows things like `fine perplexed pickle.` However, this rule has a lower chance if equally weighted with the other choices.

```
1       Noun    president
1       Noun    sandwich
1       Noun    pickle
1       Noun    chief of staff
1       Noun    floor
```

It is 1/6 in our case.
2.1.3 What values should we modify then to address the above two problems? In **2.1.1** we can decrease weight:

```
0.1    NP   NP PP
```

and in **2.1.2** we can increase weight:

```
3   Noun   Adjective Noun
```

Now after making the above modifications:

```
every president pickled every perplexed floor .
every delicious delicious sandwich wanted the sandwich .
a pickled floor in every pickle wanted a chief of staff on every floor
    !
a floor understood the pickled sandwich .
a delicious president kissed every pickle !
every pickled pickle pickled every delicious sandwich !
every delicious floor pickled every sandwich !
the floor understood a sandwich !
the floor wanted every sandwich !
a perplexed floor pickled every delicious chief of staff on a chief of
    staff .
```

2.1.4: following the above so far we can

- reduce weight for recursive formulas.
- use intuition on more common rules.
- put higher weight on verbs that are more common in English words. For instance, on `Adj`:

```
1       Adj     fine
1       Adj     delicious
0.5     Adj     perplexed
0.1     Adj     pickled
```

  Same things can be used on `Noun`, `Verb`, etc.

By the adjustments we have the following 10 sentences.

```
the delicious floor understood a sandwich .
every fine president pickled a chief of staff !
a perplexed floor on every fine president on the president ate a
    delicious president .
the pickled perplexed president understood a president .
is it true that every president ate a chief of staff ? # mixing
    terminals and nonterminals is ok.
is it true that a perplexed chief of staff understood a perplexed
    sandwich ? # mixing terminals and nonterminals is ok.
a president ate the chief of staff !
every perplexed president understood the president .
the sandwich understood every delicious sandwich .
the president with the fine floor wanted a president .
```

2.3.9 For some reason, coding in the `.gr` was hard for me!

1. Having `tab` errors, need to correct vscode so that automatically not change `tab` to `space`.
2. Below I discuss three examples and how I modified them.

Example 1.

```
the president sighed .
```

There seems to be some intransitive verbs.

Definition 2.1. An *intransitive verb* is ???

- stand alone without requiring a following NP.

Definition 2.2. A *transitive verb* is ???

- requires a direct object to indicate the person/thing receiving the action.

To address this I added.

```
#make a distinction between the two types of verbs,
0.5     VP      V_tra NP
0.5     VP      V_intra


#Add new verbs like "sighed" and "thought"
0.5     V_intra sighed
0.5     V_intra thought
```

Example 2:

```
the president worked on every proposal on the desk
```

I added

```
#the president worked on every proposal on the desk .
1       VP      V_transp PP
1       V_transp        worked
```

Example 3:

```
it perplexed the president that a sandwich ate Sally
```

Here,

- `it`, plays the rule as an *expletive*.
- This type of "that Clause" seems like a common verb pattern. However, I had to add in as a rule explicitly.

```
# it perplexed the president that a sandwich ate Sally,"
0.1     S       Expletive VP_complex
0.2     ThatS   that S
1       VP_complex      V_that NP ThatS
1       Expletive       it


#examples of V_that
1       V_that  perplexed
1       V_that  told
1       V_that  surprised
1       V_that  excited
```

2.3.10: Below I give 10 examples in `grammar3.gr`

```
a chief of staff ate the proposal !

is it true that the delicious very very very delicious president
    sighed ?
```

```
the delicious chief of staff ate the chief of staff .

the president wanted the fine proposal .

a chief of staff thought that the very very fine very delicious floor
    worked with a pickled fine floor !

the sandwich worked in the delicious president with the sandwich !

a sandwich understood the president !
the proposal sighed !

is it true that the sandwich worked under a sandwich ?

a president understood a sandwich !
```

3 Sentence ambiguity and parsing

3.1.1 The part of sentence which has ambiguity is in the PP (prepositional phrase)

- (sandwich with a pickle) on (the floor)
- (sandwich) with (a pickle on the floor)

The meaning is important in more "serious sentences". [1]

```
(ROOT (S (NP (Det a)
             (Noun (Adj pickled)
                   (Noun sandwich)))
         (VP (V_trans kissed)
             (NP (Det a)
                 (Noun (Adj fine)
                          (Noun proposal)))))
      .)
(ROOT (S (NP (Det the)
             (Noun pickle))
         (VP (V_trans wanted)
             (NP (Det the)
                 (Noun president))))
      !)
(ROOT (S (NP (Det the)
             (Noun (Adj perplexed)
                   (Noun president)))
         (VP (V_trans understood)
             (NP (Det a)
                 (Noun sandwich))))
      .)
```

[1] For example, the newsletter don't he copks killing man with knife. In the pickle example above, it doesn't matter. Both are pretty nonsense.

```
(ROOT is
      it
      true
      that
      (S (NP (ProperNoun Sally))
         (VP (V_trans wanted)
             (NP (Det the)
                 (Noun desk))))
      ?)

(ROOT is
      it
      true
      that
      (S (NP (Det a)
             (Noun chief
                   of
                   staff))
         (VP (V_transp worked)
             (PP (Prep on)
                 (NP (Det the)
                     (Noun (Adj perplexed)
                           (Noun sandwich))))))
      ?)
```

I created a `sentences.txt` file.

```
./parse -g grammar3.gr < sentences.txt > parsed_trees.txt
```

```
 (ROOT (S (NP (Det a) (Noun (Adj pickled) (Noun sandwich))) (VP (
    V_trans kissed) (NP (Det a) (Noun (Adj fine) (Noun proposal)))))
    .)
 (ROOT (S (NP (Det the) (Noun pickle)) (VP (V_trans wanted) (NP (Det
    the) (Noun president)))) !)
 (ROOT (S (NP (Det the) (Noun (Adj perplexed) (Noun president))) (VP (
    V_trans understood) (NP (Det a) (Noun sandwich)))) .)
 (ROOT is it true that (S (NP (ProperNoun Sally)) (VP (V_trans wanted)
     (NP (Det the) (Noun desk)))) ?)
 (ROOT is it true that (S (NP (Det a) (Noun chief of staff)) (VP (
    V_transp worked) (PP (Prep on) (NP (Det the) (Noun (Adj perplexed
    ) (Noun sandwich)))))) ?)
```

As of now the parsing matches pretty well.

3.2.3 This should be $C_3 = 5$ (3rd Catalan number). Since this is of the form `A with B on C under D`. Each proposition acting as a bracket.

```
    ./parse -g grammar.gr -s NP -c
```

```
    (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
```

3.3.4: I think **3.3.3** does a better job of depicting how increasing number of PP increases number of counts. The random sentences turn out to have too many "...". The example in `grammar3.gr` also turn out to be as ineffective. There was only one sentence with more than 1 parse.

```
    ./parse -g grammar3.gr -c < 3_3_4_brandom_sentences.txt > 3
       _3_4_bparse_counts.txt
```

```
 (ROOT is it true that (S (Expletive it) (VP_complex (V_that told) (NP
     (NP (Det a) (Noun desk)) (PP (Prep in) (NP (NP (Det every) (Noun
     (Adj very (Adj very (Adj fine))) (Noun sandwich))) and (NP (NP (
    Det a) (Noun proposal)) (PP (Prep in) (NP (Det the) (Noun (Adj
    perplexed) (Noun (Adj fine) (Noun pickle)))))))))) (ThatS that (S
    (NP (Det the) (Noun (Adj delicious) (Noun (Adj perplexed) (Noun (
    Adj perplexed) (Noun chief of staff))))) (VP (VP (V_trans
    understood) (NP (Det a) (Noun proposal))) and (VP (VP (V_trans
    kissed) (NP (NP (ProperNoun Sally)) and (NP (Det the) (Noun chief
     of staff)))) and (VP (V_trans ate) (NP (Det the) (Noun chief of
    staff)))))))) ?)
# number of parses = 10
 (ROOT (S (NP (ProperNoun Sally)) (VP (V_trans wanted) (NP (Det the) (
    Noun (Adj fine) (Noun president))))) !)
# number of parses = 1
 (ROOT (S (NP (Det a) (Noun pickle)) (VP (V_transp worked) (PP (Prep
    in) (NP (NP (Det a) (Noun desk)) and (NP (ProperNoun Sally))))))
    !)
# number of parses = 1
 (ROOT (S (NP (Det the) (Noun sandwich)) (VP (V_trans wanted) (NP (Det
     the) (Noun president)))) !)
# number of parses = 1
 (ROOT (S (NP (ProperNoun Sally)) (VP (V_trans ate) (NP (Det every) (
    Noun sandwich)))) .)
# number of parses = 1
 (ROOT (S (NP (Det the) (Noun (Adj very (Adj fine)) (Noun (Adj
    delicious) (Noun (Adj delicious) (Noun chief of staff))))) (VP (
    V_trans kissed) (NP (ProperNoun Sally)))) !)
# number of parses = 1
 (ROOT (S (NP (ProperNoun Sally)) (VP (V_intra sighed))) .)
# number of parses = 1
 (ROOT is it true that (S (NP (Det the) (Noun sandwich)) (VP (V_transp
     worked) (PP (Prep on) (NP (ProperNoun Sally))))) ?)
# number of parses = 1
 (ROOT (S (NP (Det a) (Noun president)) (VP (V_transp worked) (PP (
    Prep on) (NP (Det the) (Noun proposal))))) !)
# number of parses = 1
 (ROOT (S (NP (Det the) (Noun sandwich)) (VP (V_intra thought))) !)
# number of parses = 1
```

Same observation also holds: more preposition yields more parsing.

3.3.5 a)

```
    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 best parse probability is given by

$$1/3 \cdot (1/3 \cdot 1/6 \cdot 1/2) \cdot 1/5 \cdot (1/2 \cdot 1/3 \cdot 1/6)$$

$p(\text{parse}) = p(\text{sentence})$ because there is only one way to parse the sentence. As the last value is the ratio of the first two, we have 1.

3.3.5b) There are two ways to parse the sentence. Either[2]

(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 these have the same probability. Hence $p(\text{sentence}) = 2 \times p(\text{best parse})$. This is therefore why we have *exactly* 0.5.

```
    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))))
      .)
# P(best_parse) = 6.202e-10
# P(sentence) = 1.240e-09
# P(best_parse | sentence) = 0.500
```

3.3.5. c) Cross entropy is given by

$$-\log_2 p(\text{sentence})/\text{corpus size} = -\log_2(1.240 * 10^{-09} * 5.144 * 10^{-05})/18 = 2.435 \text{ bits}$$

3.3.5 d) The perplexity is $2^* = 5.4084$ where $*$ is the answer from c).

[2] I just put the obvious bracketing which makes the parsing different.

3.3.5 e) The compression does not do well because the second sentence is incomplete. Too much "surprise".

3.3.6 a My command:

```
python3 randsent.py -g grammar2.gr -n 500 | ./parse -g grammar2.gr -P
```

it has cross entropy

```
# cross-entropy = 1.712 bits = -(-8525.845 log-prob. / 4979 words)
```

Grammar 3 has more entropy than grammar 2 , since it has more variance.
The original grammar is wrong because we get "...".

3.3.7

Grammar 2 with Grammar

```
# cross-entropy = 1.933 bits = -(-10117.782 log-prob. / 5233 words)
```

Grammar 2 with grammar 2

```
# cross-entropy = 1.722 bits = -(-8579.452 log-prob. / 4983 words)
```

Grammar 2 with grammar 3

```
# cross-entropy = 2.211 bits = -(-11386.889 log-prob. / 5150 words)
```

which is what we wanted

4 Extending the Grammar

4.1.1a To address the distinction between "a" vs "an". I single out the determinants which uses `a` and `an`. That is:

```
1       NP      Det Noun
1       NP      Det_a ConsoNoun
0.2     NP      Det_a Adj Noun
1       NP      Det_an VowelNoun
```

Next, I simply copy and pasted the vocabulary of nouns and verbs and simply classify the nouns.

```
#dumb way to do this, but literally have a copy of Noun list and do
    conso and vowel noun
0.5     ConsoNoun       president
0.5     ConsoNoun       sandwich
0.1     ConsoNoun       pickle
0.5     ConsoNoun       chief of staff
0.5     ConsoNoun       floor
```

```
0.5     VowelNoun       apple
0.5     VowelNoun       elderflower
#new words the president worked on every proposal on the desk,"
0.5     ConsoNoun       desk
0.5     ConsoNoun       proposal
0.5     VowelNoun       instagram
0.5     VowelNoun       overleaf
0.5     VowelNoun       assignment
0.5     VowelNoun       university
```

But then does this address cases like `an ambivalent apple` ? Again, I would simply list out all the adjectives and put into consonant or vowel beginning words.
4.1.1b The following gives a basic framework for Yes-No q.

```
#Modify root
    1   ROOT    YesNoQ

#Yes-No questions
1       YesnoQ  Aux NP VP_base ?


#New terminal symbols

1       Aux did
1       Aux will

#possibly change eat to ate
1 VP_base eat
1 VP_base want
```