Sagiv Antebi
318159282

Ex3 - Report

1. Part 1 - Weights

1.1. The program generate so many long sentences and that's due to the use of the grammar rule:

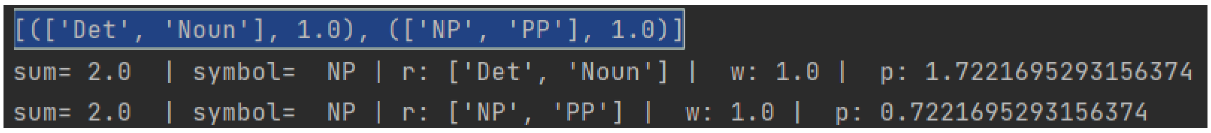
NP -> NP PP

This rule is special because it's a recursive rule, which means that unlike the other rules this rule also calls himself.
Also, we can notice that the PP rule is:

PP -> Prep NP

which means that even after using the NP rule we can also activate it again from PP rule.
For those reason we can get a sentence that is very long.

By looking at the way the code works we can understand that the number of options to pick by in this situation is 2 - so the chance is around 0.5 to always keep picking the NP -> NP PP rule option.

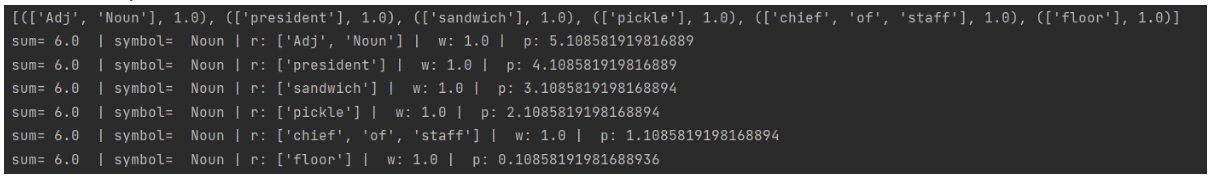


```
[(['Det', 'Noun'], 1.0), (['NP', 'PP'], 1.0)]
sum= 2.0  | symbol=  NP | r: ['Det', 'Noun'] |  w: 1.0 |  p: 1.7221695293156374
sum= 2.0  | symbol=  NP | r: ['NP', 'PP'] |  w: 1.0 |  p: 0.7221695293156374
```

In the picture we can see an example of the random p that was picked and multiplied by 2 (because of the sum) and after the first iteration the condition was true and it returned the second r (NP PP).

1.2. The grammar allows multiple adjectives, as in: \the ne perplexed pickle".
The generated sentences show this so rarely because the rules of the previous symbol is very long (is 6).

For example:



```
[(['Adj', 'Noun'], 1.0), (['president'], 1.0), (['sandwich'], 1.0), (['pickle'], 1.0), (['chief', 'of', 'staff'], 1.0), (['floor'], 1.0)]
sum= 6.0  | symbol=  Noun | r: ['Adj', 'Noun'] |  w: 1.0 |  p: 5.108581919816889
sum= 6.0  | symbol=  Noun | r: ['president'] |  w: 1.0 |  p: 4.108581919816889
sum= 6.0  | symbol=  Noun | r: ['sandwich'] |  w: 1.0 |  p: 3.1085819198168894
sum= 6.0  | symbol=  Noun | r: ['pickle'] |  w: 1.0 |  p: 2.1085819198168894
sum= 6.0  | symbol=  Noun | r: ['chief', 'of', 'staff'] |  w: 1.0 |  p: 1.1085819198168894
sum= 6.0  | symbol=  Noun | r: ['floor'] |  w: 1.0 |  p: 0.10858191981688936
```

We can see that because the number of rules is 6 than the probability to get picked is about ⅙ chance, that is much lower than the previous section (0.5 chance).
Side note: because the Weights are all 1s than i.i.d.

1.3. In order to fix 1. I need to increase the weight of the first option:

NP -> Det Noun

So I did:

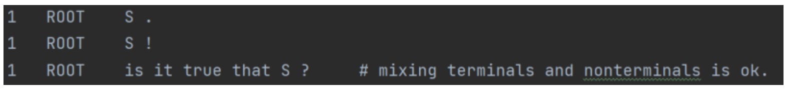
3   NP -> Det Noun
1 NP -> NP PP

Than according to this rule the chances to get: NP -> NP PP
will be ¼ .

In order to fix 2. I need to also increase the weight of the first option.
As I showed in 1.2. the reason that there is a small amount of adjectives is due to the fact that the chances are ⅙ .
By increasing the weight of the first rule (Noun -> Adj Noun), form 1 to 3, we will get that the chances now to get more adjectives is ⅜ , which is much higher than ⅙ .

By implementing those changes I got fewer NP and more Adj.
In my grammar1.gen file I saw those changes.

1.4. I noticed that the root is also i.i.d with weights of 1, but most of the sentences ends with '.' , so this rule:

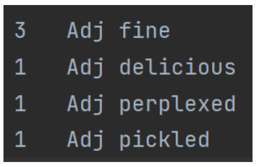


```
1   ROOT    S .
1   ROOT    S !
1   ROOT    is it true that S ?     # mixing terminals and nonterminals is ok.
```

isn't really make sense.

I changed the first rool to weight 8, so its chances will be 8/10, and the other will be 1/10.

Also at the specific word section there are some words that will suit better in sentences in others. The word 'fine' is much more suitable to be in a sentence than the word 'perplexed'. There for by increasing the weights of the word 'fine' we can get much more native like sentence.



```
3   Adj fine
1   Adj delicious
1   Adj perplexed
1   Adj pickled
```

2. Here I'll explain the changes I implemented in order to support the given constrains:

a. I added: 1   S Name VP
and also: 1 Name Sally
In order to be able to make a sentence with a reference to a specific name and then a verb

b. I added:

2   S   Name CN
2 CN Connection S

2 VP VC VP
2 VC Verb Connection

In order to support ‘and’ I needed to add a rule of CN which force a connection of add between Name and another S rule. Also for the Verb I added connection.

1 Connection and

c. I added:

3 S NP VerbNoCon

To support the rule of one verb after NP I just added this simple rule of verb without continuation.

d. I added:

4 S NP THAT
4 THAT THO S
4 THO Tho That

I added another S rule in order to allow many That - which is leagel in the English language. I splitted the THAT into THO (which will yield ‘thought’ and That that will yield ‘that’)

1 Tho thought
1 That that

e. I added:

5 S THAT1 S1
5 S1 NP VP
5 THAT1 PA That
5 PA Pronoun Adj
5 VP Verb Name

In this sentence we use ‘it’ as a pronoun so according to the sentence I added a rule that allows Pronouns and Adjective be before the sentence we want to create.
Also Added the rule S1 in order that the pronoun wo’nt repeat itself.

1 Pronoun it

f. I added:

6 NP Det VERY
6 VERY Very VERY
6 VERY Very AdjNoun
6 AdjNoun Adj Noun

As following because of the pre-adjective ‘very’ I needed to make it possible to makke multiple ‘very’ before the Adjective and the Noun, and also force to be an Adjective after the very.

1 Very very

g. I added:

7 VP Verb PNPMult
7 PNPMult PNP PNPMult

7   PNPMult Prep NP
7   PNP Prep NP

In order to make the Prep appear before the noun I needed to do these adjacents. By changing only the VP rule I forced the words before the Verb to be as the rules that are already working.After the verb we can do multiple use of the Prep rule before the noun and by allowing that we achieved our goal.

<u>**h.**</u> I added:

8   S Name CONJ
8 CONJ   Conj Adj

In order to support the sentence “Sally is lazy” - It was very easy just by adding the rule that S can do a name plus ‘is’ and then adjactive.

1   Adj   lazy
1 Conj   is

<u>**i.**</u> I added:

9   S Name CAV
9 CAV   Conj AV
9   AV   AVerb NP

In order to implement this rule I used the fact that ‘eating’ is a verb that represent a verb that happening in this moment - so it’s different from the other verbs that I currently have.
Than all I needed to add is this specific rules.

1   AVerb   eating

<u>**j.**</u> I added:

10 S Name CNP
10 CNP   Conj NP

In order to implement this rule I used the rule that I already made in d.
I just added the opthin to describe the Name with ‘is’ + NP = (det + noun).

I now explain the problem with that handling sentences (b) and (h)/(i) can interact in a bad way, to create ungrammatical sentences.
It could happens because the b. rule after the ‘and’ doing S - and than if the rules of h. or i. will be applied than it will append ‘is’ to the sentence, but its grammarly error because by using the word ‘and’ it should be ‘are’ and not is - many and not single.

4. The two additional linguistic phenomenal choose are:
(b)Yes-no questions and (d)WH-word questions.
To support those linguistic phenomenal I changed my grammar from part 2 to handle as explained below:

<u>b.Yes-no questions:</u>
I added:

20 ROOT SQ ?
20 SQ   Qus NPVP

```
20  NPVP   NP VP
```

By adding this rules, the rule that allow the first letter To be a Question form word ,and also addede the Question mark at the end of the sentence.
By giving the ROOT another option to sentence and using the regular Rules that we already ordered for NP and VP (NPVP = NP + VP).

```
# Yes No rule:
1   Qus     did
1   Qus     will
1   Qus     can
1   Qus     does
1   Qus     is
```

<u>d.WH-word questions:</u>
I added:

```
# all except 'who':
30  SQ   Wh QNPVP
30  QNPVP   Qus NPVP
30  NPVP   NP VP

# for 'who':
30  SQ   Who VP
```

By adding this rules ,the rule that allows the first letter To be a Question form word ,and also addede the Question mark at the end of the sentence.
Just like in the Yes No Question - But now I splitted it to two options: all except ‘who’ and ‘who’. and that’s because that except the word ‘who’ they all are kinda the same in terms of the sentence figure.
Wh(what) - Qus(did) NPVP(the president think) ?

but in the case of ‘who’ we dont need the Qus.

Who(who) VP(Verb(ate) NP(the sandwich)) ?

<u>**Side Note:**</u>
Because I didn’t change rules from 2, then it all works on the rules from 2.

5. I travelled the web and saw an article and in it there was this sentence:
“We’ll be able to ask them to perform tasks” - and I decided I want to give our grammar the ability to add a future wish sort of.
I added:

```
40  ROOT NP WBATRVP
40  ROOT Name WBATRVP
40  WBATRVP Wbat RVP
40  RVP RegVerb NP
```

By adding the new ROOT rule I gave the option to create first NP or Name - then the:
“will be able to” or “won’t be able to” section.

```
1   Wbat    will be able to
1   Wbat    won't be able to

1  RegVerb    eat
1  RegVerb    kill
1  RegVerb    kiss
1  RegVerb    understand
```