Kiranpreet Kaur

ECS 189G

Winter 2018

Professor Zhou

Homework 3

HW 3 Writeup

Task 1: Explain what kind of model does this PCFG implement and why?

I used the following command in the terminal given in the homework to get my output:

```
perl ./cfgparse.pl grammar2 lexicon < examples.sen
```

My output reported me three values, which are the probability of the best parse and the sentence P(T, S), the total probability of the sentence P(S), and the probability of the parse it gives you, given the sentence P(T|S). Meanwhile, it also demonstrated the best parse of the sentence in WSJ format.

We saw in Chapter 4 a simple approximation of this probability using N-grams, conditioning on only the last word or two instead of the entire context; thus, the bigram approximation would give us P(wi |w1,w2,...,wi−1) ≈ P(wi−1,wi) P(wi−1).

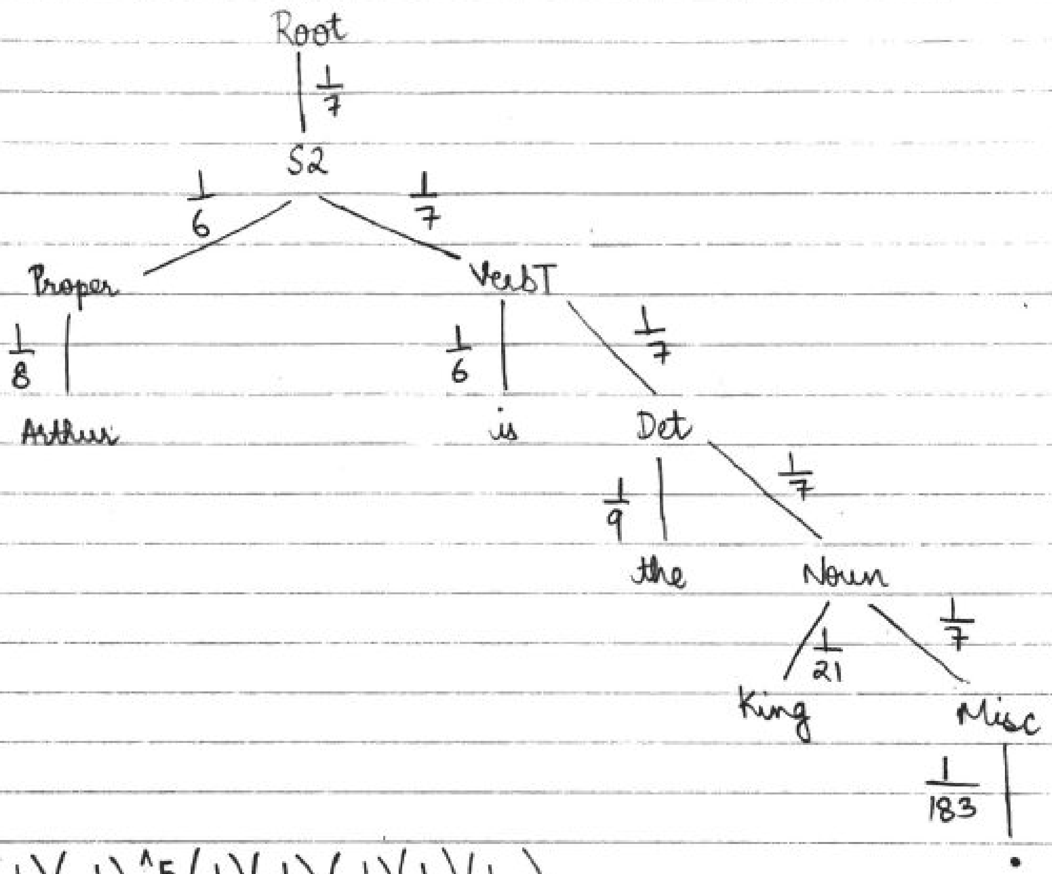
I guessed that the PCFG implements the bigram language model. Then, I performed some calculations to confirm if PCFG really does implement the bigram language model.

P(T,S) = 1/6*(1/7)^5*(1/8)*(1/6)*(1/9)*(1/21)*(1/183) = 5.973*10^(-12)

P(S) = 0.500

P(T|S) / P(S) = 1.195 x 10^-12 (which is same as the probability in the code)

Therefore, this confirms our initial guess that PCFG implements bigram language model.



$P(T|S) = \left(\frac{1}{6}\right)\left(\frac{1}{7}\right)^{\wedge}5\left(\frac{1}{8}\right)\left(\frac{1}{6}\right)\left(\frac{1}{9}\right)\left(\frac{1}{21}\right)\left(\frac{1}{183}\right)$

$= 5.973 \times 10^{-12}$

We Know, $P(S) = 0.500$

<u>Formula</u>

Bigram model: $Pr(T|S) = \frac{Pr(T|S)}{P(S)}$

$$= \frac{5.973 \times 10^{-12}}{0.500}$$

$$= 1.195 \times 10^{-11}$$

Task 2: Compare two outcomes and explain the differences between them.

I generate outcome 1 (grammar1 file) by the following command in the terminal:

```
perl ./cfgparse.pl grammar1 lexicon < examples.sen
```

I generate outcome 2 (grammar1 file + grammar 2 file) by the following command in the terminal:

```
./cfgparse.pl grammar1 grammar2 lexicon < examples.sen
```

Outcome 1 showed failure after displaying probabilities and grammar in WSJ format in the first two lines. However, outcome 2 showed no failures in any line and gave the probabilities as well as the grammar in WSJ format (with a + sign in front of every grammar) of the the whole file.

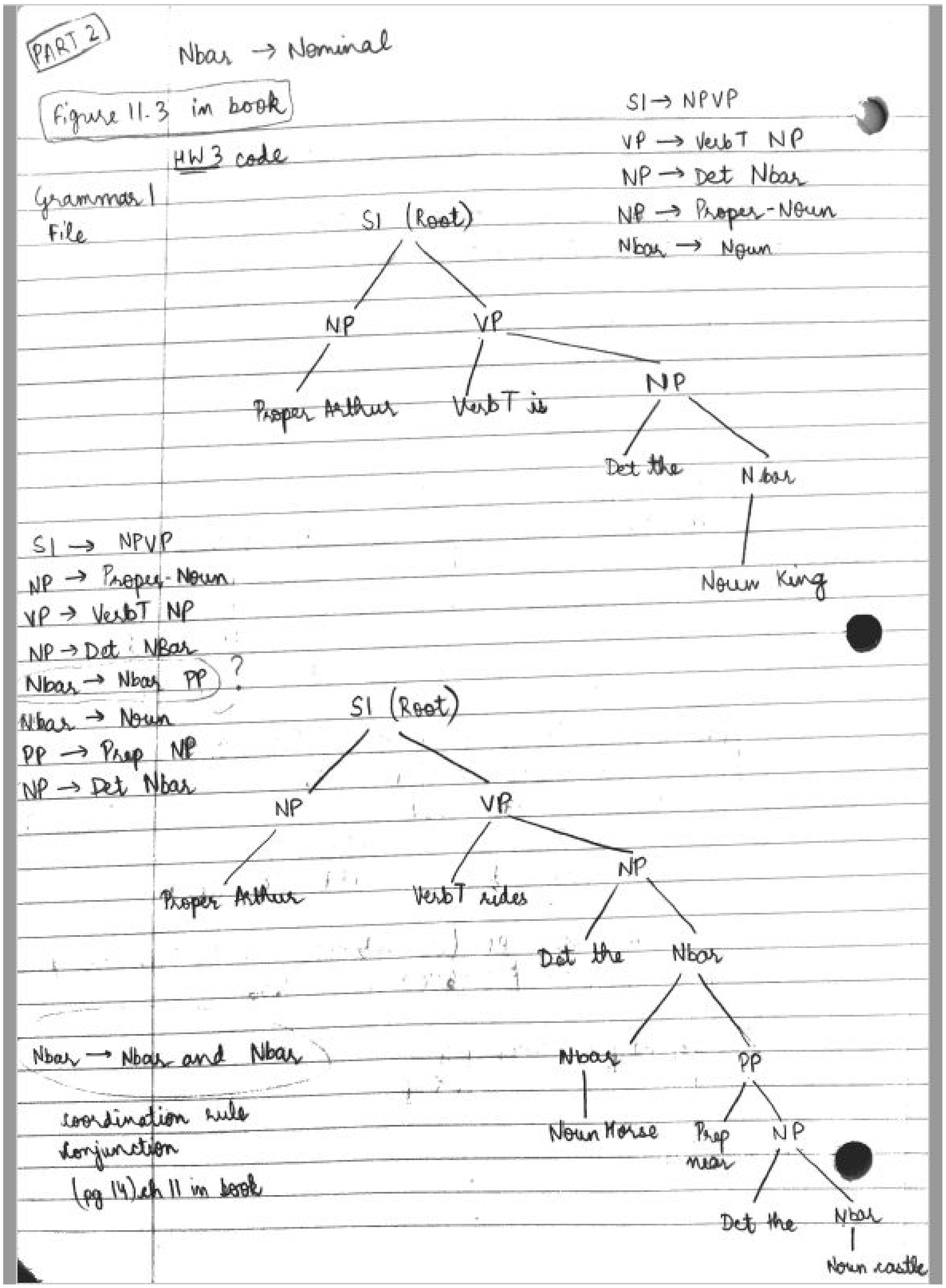
The lexicon file consists of words that have tags such as Noun, Det, Prep, Proper, VerbT, and Misc. These tags are important since they form the grammar rules. These tags are the only way to lead to these words.

The cfgparse.pl can take more than one grammar file on the command-line; it just merges the rules together and sums up each rule's weights if they occur more than once. Thus, when we get outcome 2, it merges the rules in grammar1 and grammar2 file and gives us the output of the whole file. A variety of input can be interpreted as different grammar sentences because of the flexibilty in the rules. The grammar2 file takes care of the Misc words in the lexicon file while grammar1 does not which is why we get errors in the Outcome 1. This also explains when we use grammar2 and grammar1 together, we get the total output of the whole file without any errors.

In our Outcome 2, we have some non-terminals starting with a "+" sign which makes us treat them differently than the other non-terminals without a "+". For example, +Det is different from Det, but yes +Det can be treated as one type of terminal, it can be replaced by other symbols like X, Y, Z, etc. However, Outcome 1 does not have any "+" sign non-terminals. So, we treat everything as the same.

Any sentences that can only have one unique parse will always result in a P(S) = 2*P(S,T), and the P(T|S) will be 0.5 from the code. Thus, whenever you see P(T|S) = 0.5, you can safely assume there is only one parse for the sentence. If the sentence has multiple parses, then P(T|S) will be less than 0.5. In both our outputs, outcome 1 and outcome 2, the P(T|S) = 0.5 so we can assume that there is only one parse for the given sentences.

Parse trees for Outcome 1 from the grammar1 for the output displayed on terminal:



PART 2

Nbar → Nominal

figure 11.3 in book

HW3 code

Grammar 1
File

SI → NPVP
VP → VerbT NP
NP → Det Nbar
NP → Proper-Noun
Nbar → Noun

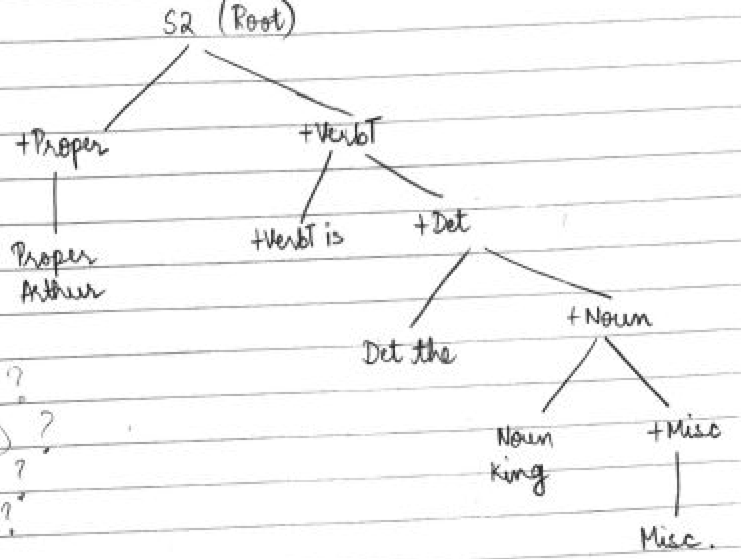
SI → NPVP
NP → Proper-Noun
VP → VerbT NP
NP → Det NBar
Nbar → Nbar PP ?
Nbar → Noun
PP → Prep NP
NP → Det Nbar

Nbar → Nbar and Nbar

coordination rule
conjunction
(pg 14) ch 11 in book

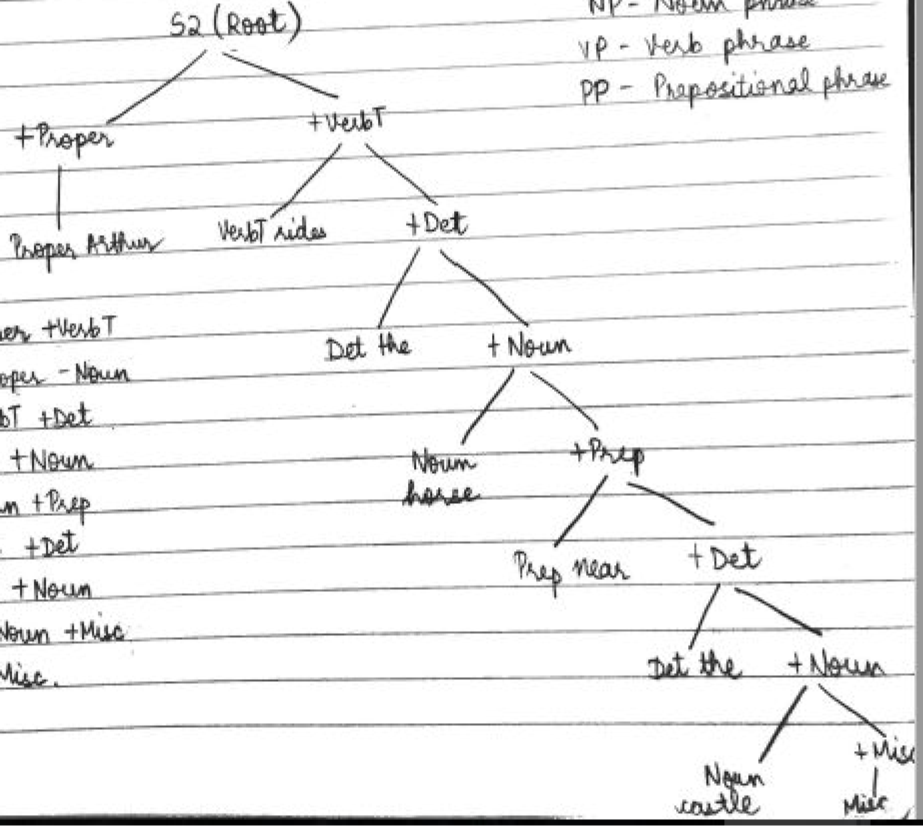
Proper, Noun, Det, VerbT, Prep are defined in Grammar2 file

Grammar2 file



S2 → +Proper +VerbT ?
+Proper → Proper-Noun ?
+VerbT → +VerbT +Det ?
+Det → Det +Noun ?
+Noun → Noun +Misc ?
+Misc → Misc.

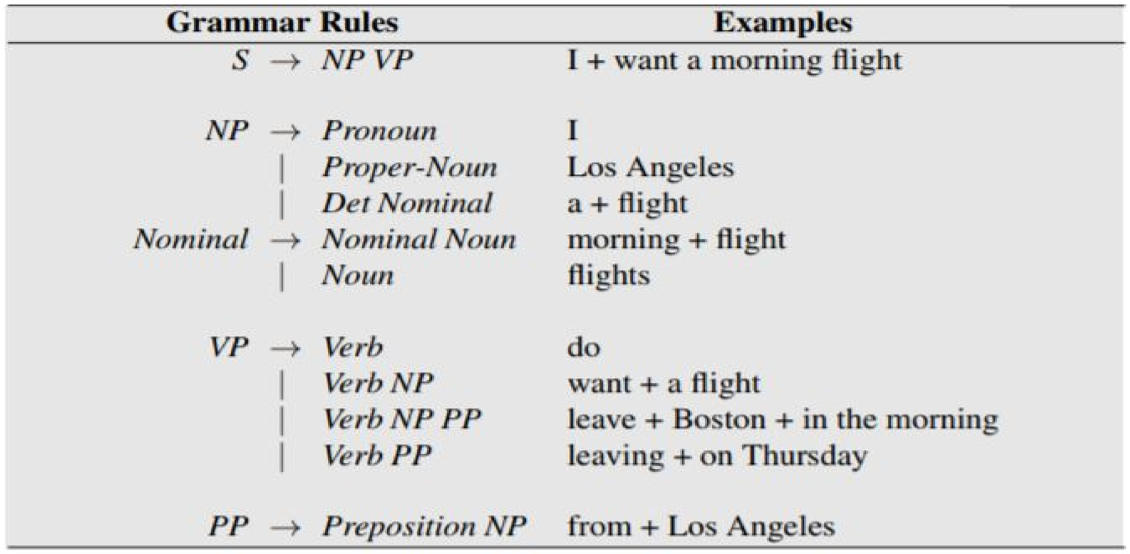
NP - Noun phrase
VP - Verb phrase
PP - Prepositional phrase



S2 → +Proper +VerbT
+Proper → Proper -Noun
+VerbT → VerbT +Det
+Det → Det +Noun
+Noun → Noun +Prep
+Prep → Prep +Det
+Det → Det +Noun
+Noun → Noun +Misc
+Misc → Misc.

The output shows that the rules in grammar1 file are incomplete as they do not take care of the tag “Misc” for some words that can only be accessed through these tags when we run it alone on the examples.sen file, it results in failure for most part of the file. When I drew out the parse tree for grammar1 file, I discovered that all the grammar rules were consistent with the formal grammar rules given in the book in Figure 11.3 except for one rule Nbar(Nominal) -> Nbar(Nominal) PP. Therefore, this rule gives ungrammatical sentences which shows as “failure” in our terminal. As the book explained, sentences that cannot be derived by a given formal grammar are not in the language defined by that grammar and are referred to as **ungrammatical**.

Formal grammar rule table from the book:



Grammar Rules			Examples
S	→	*NP VP*	I + want a morning flight
NP	→	*Pronoun*	I
	|	*Proper-Noun*	Los Angeles
	|	*Det Nominal*	a + flight
Nominal	→	*Nominal Noun*	morning + flight
	|	*Noun*	flights
VP	→	*Verb*	do
	|	*Verb NP*	want + a flight
	|	*Verb NP PP*	leave + Boston + in the morning
	|	*Verb PP*	leaving + on Thursday
PP	→	*Preposition NP*	from + Los Angeles

Figure 11.3 The grammar for $\mathscr{L}_0$, with example phrases for each rule.

Task 3: Compare three outcomes and explain the differences between them.

I generate outcome 1 (grammar1 file) by the following command in the terminal:

perl ./cfggen.pl --text 50 grammar1 lexicon

I generate outcome 2 (grammar2 file) by the following command in the terminal:

perl ./cfggen.pl --text 50 grammar2 lexicon

I generate outcome 3 (grammar1 file and grammar2 file) by the following command:

perl ./cfggen.pl --text 50 grammar1 grammar2 lexicon

N should be an integer which indicates number of sentences to be generated. I generated 50 sentences and compared my outcomes.

The outcome 1 is based on grammar1 file which has the grammar rules from proper tags and there are no Misc tags. This makes our output very uniform as the sentences are obtained by applying these specific tags. The length of each sentence is pretty short (usually 5 words) and very uniform. The ouput produces meaningful sentences with good context.

The outcome 2 is based on grammar2 file which has the grammar rules from various tags including the Misc tags. This makes our output very scattered as the sentences are obtained by applying all sorts of tags and giving a lot of flexibility by using Misc tags. The length of each sentence is not uniform (ususally ranging from 1 word to 24 words). The output produces absurd sentences with bad context. Sometimes, it randomly places a word individually and sometimes in a long sentence.

The outcome 3 is based on grammar1 and grammar2 file. Here, it adds the rules of both grammars and brings us an output which is very similar to the outcome 1. The sentences are pretty uniform and short. As far as the context behind these sentences is concerned, nothing specific could be said about it because it is a mixture of meaningful as well as absurd sentences. Also, we can see that grammar1 rules are given more priority since it has a bigger weight on the ROOT (99) as compared to the ROOT (1) in grammar2 file. We can confirm this by seeing that there is a period "." at the end of each sentence in our output for the merged grammar which shows the priority of rules of grammar1 file.

Task 4: Build your own PCFG grammar!

For this task, I created two files mygrammar and mylexicon.

The mygrammar file defined a set of PCFG rules to derive any non-terminal symbols to other non-terminal symbols with the format: Nonterminal_A -> Nonterminal_B weight. In order to define these PCFG rules, I used the ones defined in grammar1 and grammar2 file, combined them and modified the weights assigned to be 1 for all the rules. I also added rules for EOS (End of sentence) tags, such as "+Noun -> Noun EOS 1", "+Proper -> Proper EOS 1", etc. for all the tags.

The mylexicon file defined a set of rules to derive any non-terminal symbols to terminal symbols with the format 'non_terminal -> terminal weight'. The weight after each rule represented the probability of deriving the righthand side from its lefthand side. Also, we could use the words from the file wordlist for terminals and never adding any new words in mylexicon file. I also changed the tags for characters like ".", "!", "?", etc from Misc to EOS (End of sentence) tags.

The grammar made by me should ideally assign high probability to any grammatical sentences and very low probability for non-grammatical sentences. Also, it should never fail when parsing any string of words. Initially, mygrammar file was giving me some failures for some string of words. **What did I do to fix the issue of failure in some sentences?** There were three failures in my sentences. I compared these sentences to the original output given bby lexicon file and figured out that my output was failing because I changed the tag of a "," to EOS. However, a "," is not the end of sentence, it can come in the middle of the sentence too. Similary, ";" and ":" are not end of sentences. So, I changed their tags back to Misc. Now, my output did not show any failure for any of my sentences.

I used entropy to evaluate the performance of mygrammar. I made a test script code entropy.py in python and tested the output. I stored the output in a txt file for outputs given by merged grammar (grammar1 and grammar2) and mygrammar. I got a score of **72.3603515886** for the output generated by merged grammar (grammar1 and grammar2) and a score of **60.2419383258** for the output generated by mygrammar.

Task 5: Use cfggen.pl to generate 20 sentences from mygrammar !

I used the following command in the terminal to generate 20 sentences from mygrammar:

```
perl ./cfggen.pl --text 20 mygrammar mylexicon
```

Then, I stored these 20 sentences in a file called generated.txt file. **What did I do remove all the ungrammatical sentences?** I manually checked my generated.txt file ad used human knowledge to remove all the ungrammatical sentences and saved the output in a new file called generated-grammatical.txt file.

My generated-grammatical.txt file had 7/20 correct sentences.

References

Book: Online new chapters from Jurafsky and Martin. third edition in progress. Speech and Language Processing. Link provided: http://web.stanford.edu/~jurafsky/slp3/

http://web.stanford.edu/~jurafsky/slp3/11.pdf

https://nlp.stanford.edu/~socherr/CGWDocumentationCS224N

https://catalog.ldc.upenn.edu/ldc2000t43

http://anthology.aclweb.org/H/H92/H92-1073.pdf

Piazza page for this course was really helpful!