ECS 189G NLP HW3: Probabilistic Context Free Grammar

Name: Sairamvinay Vijayaraghavan, Student ID: 913603345

May 22 2019

1 Task 1

Based on the outputs of the cfgparse.pl script and the grammar rules employed, it can be deduced that the model is a simple bigram Hidden Markov Model (HMM). This conclusion was made based on the following justifications:

Every grammar in grammar 2 was in its CNF form where each Non-Terminal breaks into either 2 non-terminals or one single terminal.

The grammar2 can hence be simplified into the following structure:

$root \rightarrow S2$

$S \rightarrow empty| + (NT)$

$+(NT) \rightarrow NT + (NT)|NT + (NT2)$

$NT \rightarrow Terminal$

$NT2 \rightarrow Terminal$

Here +(NT) refers to the state to which a sentence can transmit to. For example in the sentence "Arthur is a king.", The Root breaks into Sentence terminal S. Then S breaks into the +(Proper) non terminal where the tree keeps breaking down to yield the final parse tree.

According to this model, while comparing with HMM, the states are the tags to which the tokens are finally being tagged to. Emission probability can be viewed as the probability of the observation (the word) from the given state (the tag in which the word is associated with). The probability for the transmission can be interpreted as $P(tag2|tag1) = Count(tag1 \rightarrow tag2)/Count(tag1)$

Clearly, this is the transmission probability for a HMM model where the probabilities are calculated based on the count.

Similarly the emission probability of an observation W_i given the tag T is given by:

$P(W_i|T) = Count(W_i, T)/Count(allwordsWinT)$

In these calculations, it can be viewed that the probability of generating the parse tree T and the sentence S, is the product of the transition probabilities and the emission probabilities.

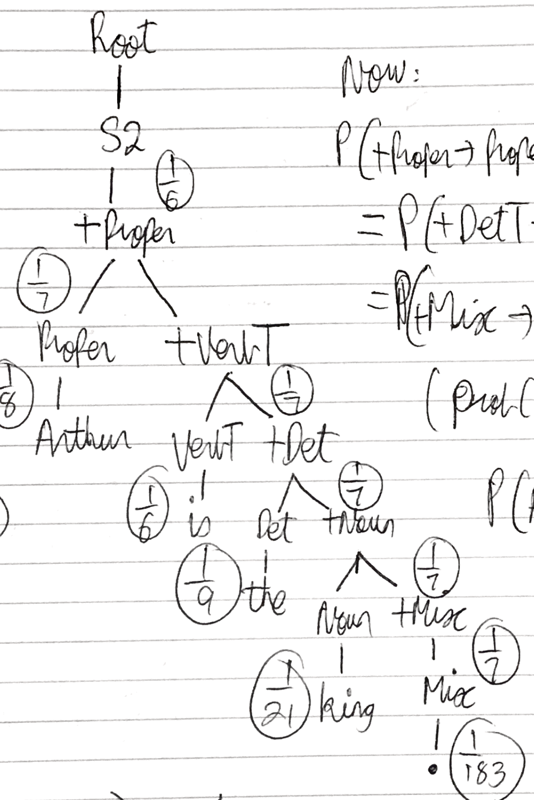
The following diagram (inside the picture) explains our verification of this model used as being a HMM by finding the P(Parse Tree T, Sentence S) where the Sentence S = "Arthur is the king." as per grammar2 rules.

As we can see, the Markovian property can be observed in the transition probabilities. The current state depends only on the previous state, and this can be seen in the calculations below.

Sentence = "Arthur is the king."

let's assume a HMM and then try to verify

The parse tree for this sentence given by the program is as follows: (This has maximum probability parse tree):-



$P(\cdot \mid \text{Mix}) = \frac{1}{183}$

Now:

$P(\text{+Proper} \to \text{Proper +VerbT}) = P(\text{+VerbT} \to \text{VerbT +DetT})$

$= P(\text{+DetT} \to \text{Det+Noun}) = P(\text{+Noun} \to \text{Noun +mix})$

$= P(\text{+Mix} \to \text{Mix}) = \frac{1}{7}$

(Prob (any NT breaking into 2 other NT or single terminal) (except S_2))

$P(\text{Arthur}/\text{Proper}) = \frac{1}{8}$

$P(\text{is} \mid \text{verbT}) = \frac{1}{6}$

$P(\text{the} \mid \text{Det}) = \frac{1}{9}$

$P(\text{King} \mid \text{Noun}) = \frac{1}{21}$

Now $P(\text{+Proper} \mid S_2) = \frac{1}{6}$ (because out of 6 non-empty breakdowns for S_2, only 1 leads to +Proper)

~~$P(\text{Proper} \mid \text{+Proper}) = \frac{1}{7}$ $P(\text{+VerbT} \mid$~~

3

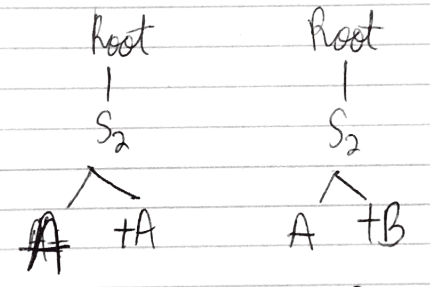
$P(\text{Parse Tree T and Sentence S}) = P(\text{all transitions}) \cdot P(\text{all emission})$

$$\boxed{P(T \cap S)} = \underbrace{\left[\left(\frac{1}{7}\right)^5 \cdot \frac{1}{6}\right]}_{\text{Transition}} \cdot \underbrace{\left[\frac{1}{8} \times \frac{1}{6} \times \frac{1}{9} \times \frac{1}{21}\right] \times \frac{1}{183}}_{\text{emission}}$$

$$= \boxed{5 \cdot 973 \times 10^{-12}}$$

Here there are totally ~~5~~ 5 states (Terminals) for this sentence.

Now: With this grammar 2 types of parse Trees can be obtained (broadly)



$$\therefore P(\text{Parse Tree } T \mid \text{Sentence } S) = \frac{1}{2} = 0 \cdot 5$$

$$\boxed{\text{Here } P(S) =} \frac{P(T \cap S)}{P(T \mid S)} = \frac{5 \cdot 973 \times 10^{-12}}{0 \cdot 5} = \boxed{1 \cdot 195 \times 10^{-11}}$$

Hence we have verified that P(S) calculated is the same as the result given by program

Clearly the final result of using the markovian property explains and confirms our conclusion of that the probabality of T,S found by math is the same as the probability calculated by the perl script. Hence this model is a bigram HMM.

2 Task 2

Interestingly, when the grammar 1 file was alone ran, the output showed that the parsing was a failure for most of the sentences, while it was successful for only the first two sentences.

However, when the grammar 2 and grammar 1 files were combined in running the parse script, all the 20 sentences were parsed and the probabilities were reported along with the parsing trees. Of course, the probabilities in this output are different from the output of task 1 which only used grammar 2.

The lexicon file contained the following tags and marks their emissions with some weights (possibly counts) which contains the emissions for these tags: Noun, Det, Prep, Proper, VerbT, and Misc, with most of the tokens falling under the category of Misc.

The output when only using grammar 1 was very unsuccessful while parsing was because of the very few rules in grammar1. For example, the Misc tag was completely missing and surprisingly, 183 words (tokens) fall under the Misc tag as described in the lexicon file and similarly other rules such as $S1 \rightarrow NPVP.$ leads to a break down of S1 into two non-terminals and a terminal (period) which does not seem to be following the Normal form of compressing the grammar rules.

Also, most of the sentences have tokens which fall broadly under the category of Misc. Since, this state was completely absent within grammar 1 rules, almost 90% of the sentences were failure in parsing. The script could not be able to parse the entire tree as most of the delimiters fall under Misc state.

However, in case of parsing when combining both the grammar rules, we can see that now Misc is included (in grammar 2) and hence it gets accounted while parsing the same tokens from the same sentence file and hence parsing is now successful for most of the sentences.

3 Task 3

I had generated around 100 sentences based on each form of grammar in each to gauge its outcomes

Now after the script have been run on the three forms of grammar, the results are written as follows:

3.1 Grammar 1

Based on the grammar 1 output, we can find that this grammar basically parses a sentences based on parse chunking. Since this grammar uses only a very few set of tags which have short rules and mainly non-recursive Non terminals (for the most of the terminals), the sentences generated were short and were uniform (can be seen that every sentence ends with a period as the main rule for any sentence is only one rule $S \rightarrow NPVP$. and hence every single sentence starts with a noun phrase and followed by a verb phrase and always ends with a delimiter.

This grammar provides more of a overall structure to any sentence and hence it provides more grammatical sentences, with the same general sentence structure.

3.2 Grammar 2

Interestingly, grammar 2 was not providing quite meaningful sentences and hence were more ungrammatical. This can be inferred because in grammar 2, the model used a bigram HMM and since every tag depends solely on its previous tag, the sentences formed are completely random in length and in sentence structure. Hence, in comparison to grammar 1, there is a complete lack of grammatical structure overall, which hence provides more ungrammatical sentences than more meaningful sentences.

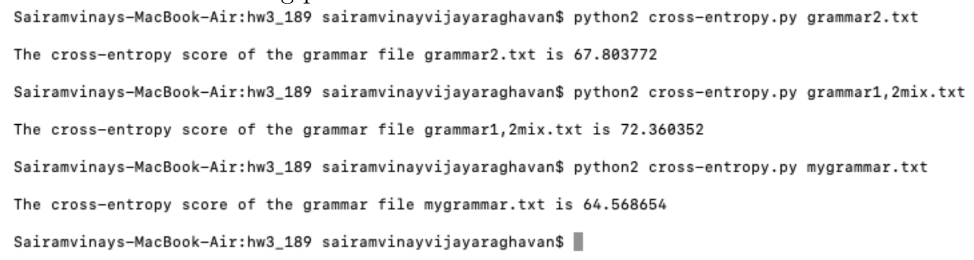
3.3 Mix of grammar 1 and 2

Finally, when we try to combine both the grammar rules, it can be observed that the output is indeed a mix of both grammatical and ungrammatical sentences, but however the number of grammatical sentences outweigh the number of ungrammatical sentences.

This can be explained based on the weightage inside the grammar files. It can be observed that 99% of the weight is emphasised for the grammar 1 basic structure $Root \rightarrow S1$ while only 1% for the grammar 2 structure. This explains the resemblance of the output with the output of the sole grammar 1 case. Hence this combinations provides more grammatical sentences than the sole grammar 2 model but lesser than the sole grammar 1 model.

4 Task 4

In order to design my own grammar which was required to do better than the merged version of grammar 1 and 2, I had designed my grammar accordingly. First, I had retained most of the rules of grammar 2 in my grammar design. I had done this, because grammar2 ensured a more successful parse than grammar 1 and hence grammar 2's rules were mostly retained except with some changes. Then, after noticing that majority of the tokens were belonging to the "Misc" tag. I had then added some more states under the Non-terminal "Misc", which had now had decomposed into the Non-terminals (which lead to the final tokens), namely CC (for conjunctions like but, and, so, neither, either, or), END (for the end of sentence symbols (?!.)), Punct (for the punctuation symbols :,;) , Adj (the adjectives such as tropical, sacred, hot, lucky) and Adv (for adverbs like trusty,already etc.) and Qn (Why, What, How, When, Which), NNP (for plural nouns), VBP (for the verbs in past tense) and also finally the Extra tag, which accounted for the rest of the words (which were not been able to be tagged under these tags). The weights were assigned accordingly for all of these $Misc \to NT$ rules with the most probable tags (such as Extra and End (as most of the words fall under those categories and those words are almost present in every sentence), with high probabilities (weights) while less grammatical sentences having a lower probability. Similarly mylexicon file has no additional words besides the words in the given wordlist file, but some of the words were now assigned to the specific tags with different weights. For example tokens such as period (.) was assigned a very high probability (for grammatical reasons) in the END tag, while certain uncommon tags such as 's inside Extra tag and some non-frequents adjectives such as angolian were assigned a very small weight which made it highly unlikely as it was non-grammatical. My grammar and lexicon files did not fail any of the example sentences. The cross-entropy scores of each of the grammar files (2, mix of grammar 1 and 2, mygrammar) were recorded as in the following picture:



```
Sairamvinays-MacBook-Air:hw3_189 sairamvinayvijayaraghavan$ python2 cross-entropy.py grammar2.txt

The cross-entropy score of the grammar file grammar2.txt is 67.803772

Sairamvinays-MacBook-Air:hw3_189 sairamvinayvijayaraghavan$ python2 cross-entropy.py grammar1,2mix.txt

The cross-entropy score of the grammar file grammar1,2mix.txt is 72.360352

Sairamvinays-MacBook-Air:hw3_189 sairamvinayvijayaraghavan$ python2 cross-entropy.py mygrammar.txt

The cross-entropy score of the grammar file mygrammar.txt is 64.568654

Sairamvinays-MacBook-Air:hw3_189 sairamvinayvijayaraghavan$ 
```

As seen from the picture, the combined grammar of grammar1 and grammar2 had a cross entropy score of 72.360352 and the grammar2 file alone gave a score of 67.803772 while mygrammar had beaten both of these grammar files with a score of 64.568654.

5 Task 5

The mygrammar and mylexicon was run with cfggen.pl script to yield 20 sentences based on mygrammar. The output was generated and the grammatical sentences was just observed and isolated manually.

I got 6/20 sentences to be grammatical.