**CSCI 544 Applied Natural Language Processing | Fall 2017**

**HW Assignment 4 – Parsing**

**Harshith Guru Prasad | USC ID: 2735 0858 94**

**ANSWER 1**

The number of rules in the grammar is **752.**

The most Frequent rule : **PUNC -> ‘.’**

Frequency : **346**

*Code present in Build\_pCFG.py*

**ANSWER 2**

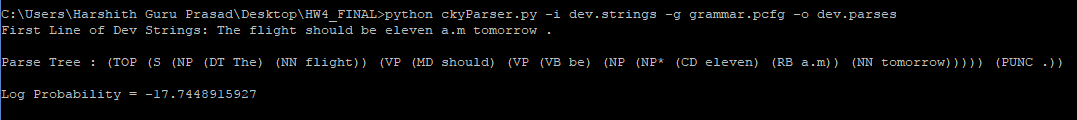
Dev.string [0] – “The flight should be eleven a.m tomorrow .”

Parser generated result

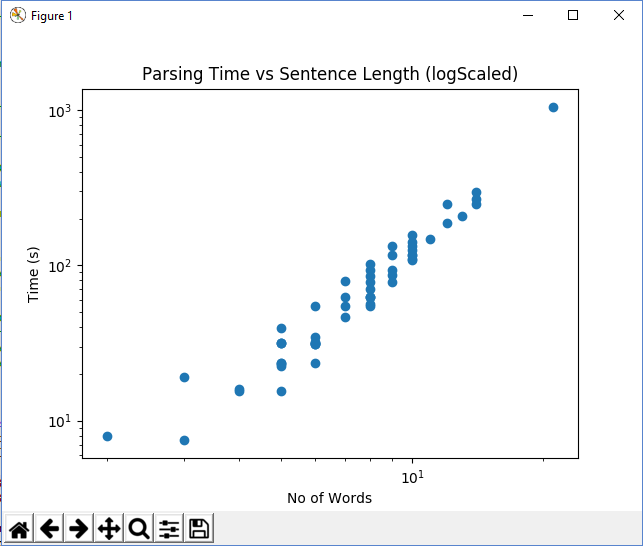
(TOP (S (NP (DT The) (NN flight)) (VP (MD should) (VP (VB be) (NP (NP\* (CD eleven) (RB a.m)) (NN tomorrow))))) (PUNC .))

p=1.79932e-18

math.log(p=1.79932e-18) = -17.74892



**ANSWER 3**



After performing a **linear regression** and the **least square fit** method using python’s

**Scikit-learn(sklearn)** and **pyplot** library functions to minimize the least squared errors,

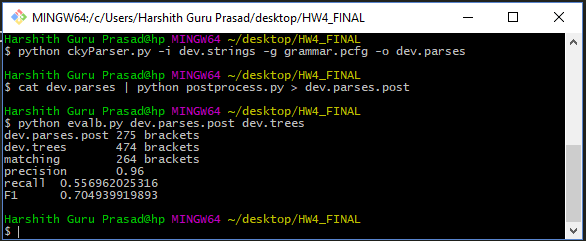
I approximated the value of k to be nearly 2.79. The value of k is close to three as the

**runtime complexity of the CKY parsing algorithm is cubic** with respect to the number

of words in the sentence. CKY parsing is of the order of n to the power of 3(cubic runtime).

Hence **k = 2.79** and is quite **close to 3** where y = x ^ k.

**ANSWER 4**

****

**Precision = 0.96**

**Recall = 0.55692025**

**F1 - score = 0.7049399**

**ANSWER 5**

To improve/enhance the F1 score / performance of the CKY parser, I implemented the standard practices of replacing rare/infrequent words having a frequency less than 2 with <unk>. Preprocessed the train.trees file to binarize and removed unary relations between nodes. Used log probabilities to prevent underflow. Used a training dictionary of words with a frequency more than 1 to identify sparse words and replace them with <unk>. Recorded the replaced words so that they can be used to replace the <unk> tag at the time of post processing of the syntax trees formed by CKY parsing of the test sentences.

**Modification 1**

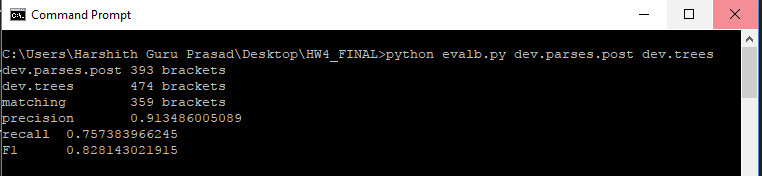
**Smoothing** for lexicalised pCFG – replaced infrequently seen/used words with <unk> when parsing and replaced some interesting infrequent words with stems like <unk>-ing. Improved the F1 score by 4-6%. Used the snowball stemmer to stem infrequent words with interesting suffixes to increase the overall parse recall thereby leading to a better F1 Score. Tried head lexicalising and binarization. Achieved slightly better results.

**Modification 2**

**Parent annotation:** Appended the label of the parent to each of its binarized dependents. Provides more specific rules to parse the sentence and is simultaneously not too specific that it cannot generalize well to new sentences/words. Improved the accuracy by 3-5%. Tried to annotate sibling information into the nodes at the expense of a quadratic runtime complexity that made the model more prone to overfitting.

**Modification 3**

**Beaming:** Tried to enhance the parser’s F1 score by neglecting all the states that have the least probabilities. Once a higher probability is found, the beaming procedure eliminates the states having lower probabilities including their back tracking records. However, it did not affect the F1 score by the expected measure and it runs the risk of returning a suboptimal solution and leads to search errors for some inputs. Made other slight variations to achieve a F1 score of nearly 0.83.

****

