Name : Raka Dalal
Campus Id : JG68695

methods.py and 1_2_3.py are the codes for Question 1, 2 and 3.

1a.

Total number of sentences in training split: 151401
Average number of sentences in each training document: 30
Total number of sentences in development split: 60166
Average number of sentences in each development document: 30

I am surprised at the average because both in training and development split, there are some documents which are huge and there are some documents which are quite small but when we compute the average, all the documents have a standard size.

1b.

Total number of tokens in training split: 2424068
Total number of different word types in training split: 125553

1c.

The most common words are punctuation, determinants, prepositions and conjunctions.
One way of grouping these words together is converting all words to lower case. For example ‘The’ and ‘the’ will be one single word then. Also we can remove all punctuation marks from the corpus.
One disadvantage of such grouping technique is that if words of different cases convey different meanings then those meanings will be lost. Also punctuation marks convey a lot of meaning in sentiment analysis which will be lost.

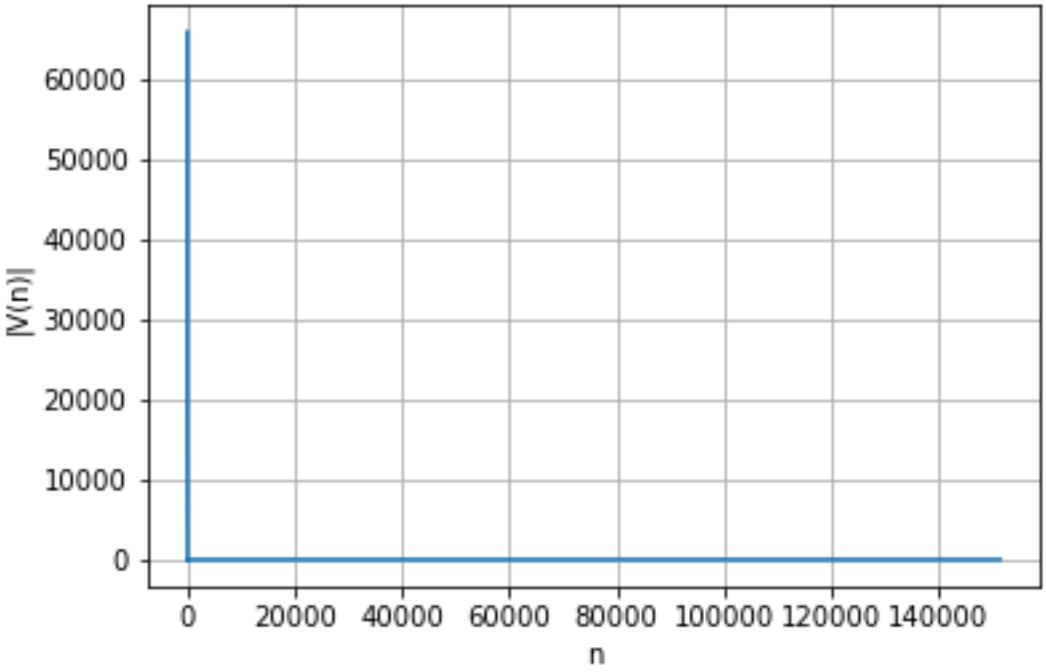
1d.

The least common words are the rare words present in a corpus. If we want to create a language model we will eliminate these words from the corpus as they are less likely to occur in some other corpus. But these rare words will be a valuable source of information in case of information retrieval or sentiment analysis.

1e.

Total number of tokens in development split: 938493
Total number of different word types in development split: 69198

2a.



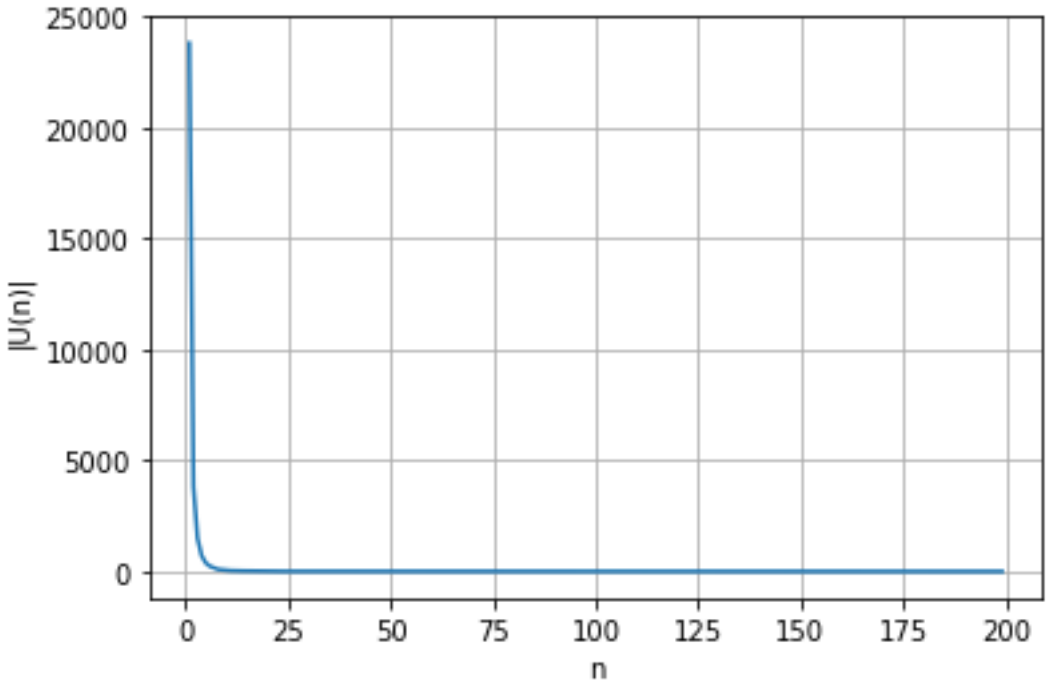
It's quite surprising that words of very less frequency are huge in number and vice versa.

2b.

Number of out of vocabulary words:31119

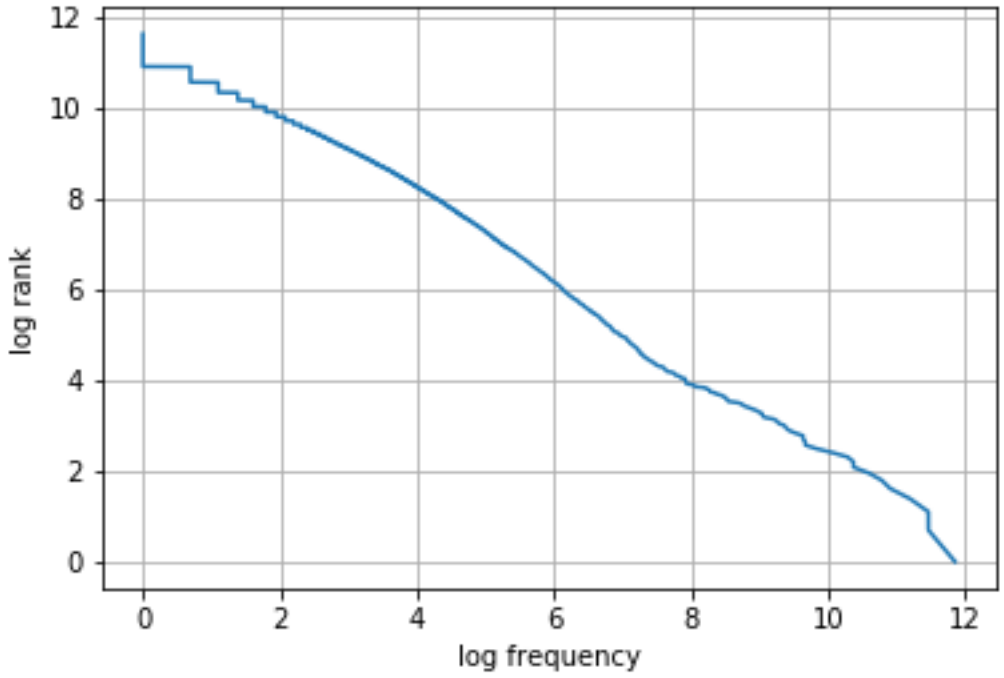
The same observation holds here. Words of very less frequency are huge in number and vice versa.

3.



I
have used four configurations.

First configuration(word transformation:None, amount of data used: 100%):

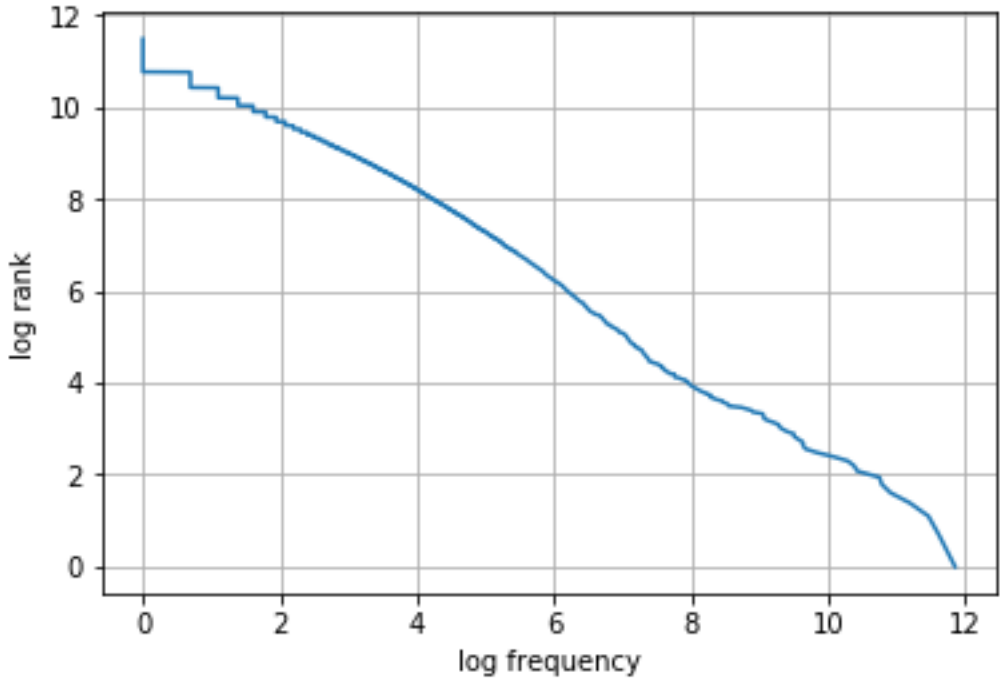


Here, slope = -0.78072210956
R^2 = 0.97131337

log rank = -log frequency + log C
Ideal value of slope should be -1. So, this computed value of the slope stems from the noise in the real dataset but even then the value is quite close to -1 which shows that Zipf's law holds empirically in this case.

R^2 is a statistic that gives information about how well the model fits the data. The more the value is close to 1, the more it indicates that the variance in the dependent variable is predictable from the independent variable. Hence the R^2 value of 1 means a perfect fit for the data. The fact that the R^2 value in this experiment is very close to 1 shows that the noise is very low in the data further strengthening the claim in Zipf's law.

Second configuration(word transformation: lower case, amount of data used: 100%):

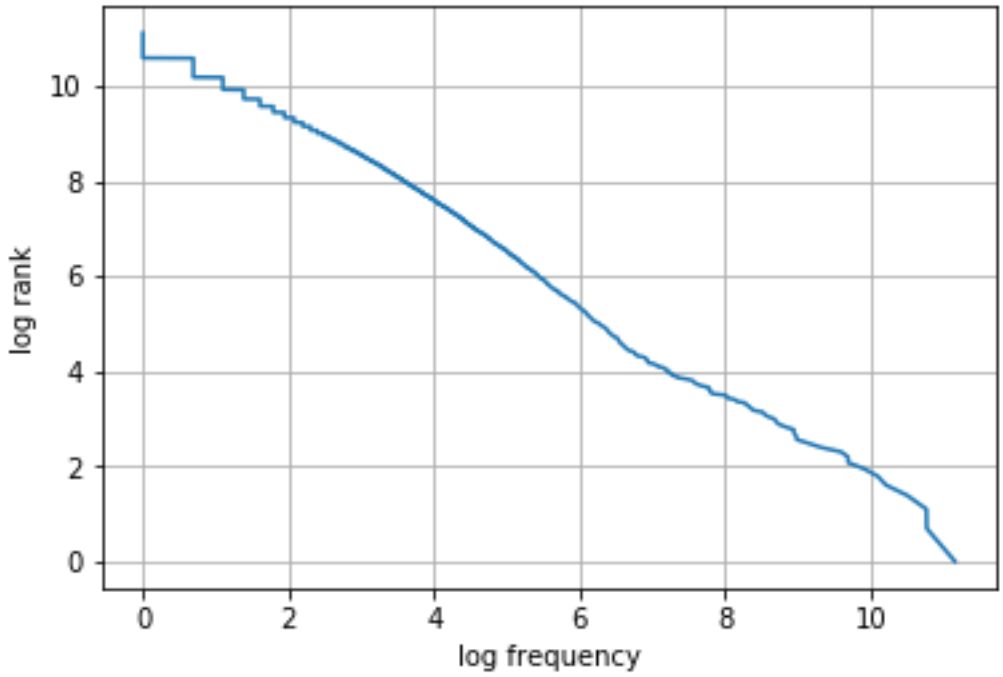


Here, slope = -0.762700742616

$R^2 = 0.971587809575$

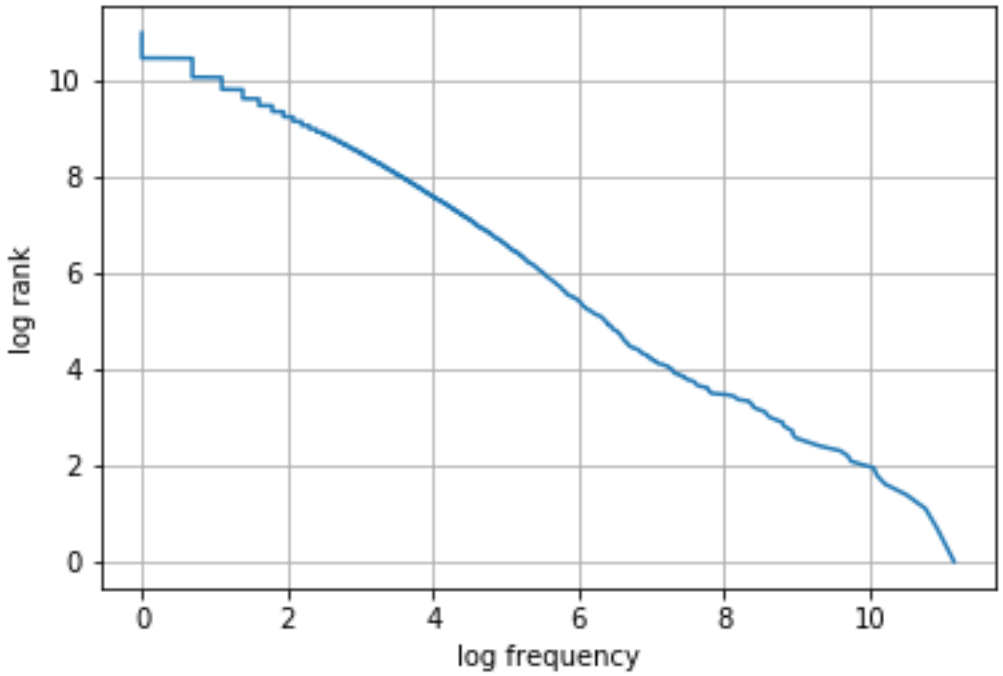
Here, the slope and R^2 value is almost same as the previous one. So, the same explanation holds here.

Third configuration(word transformation: None, amount of data used: 50% sampled uniformly at random):



Here, slope = -0.821709847377
R^2 =0.980181743775

Here, the value of slope is more close to -1 and value of R^2 is more close to 1. So, it can be said that in this experiment the model fits better.



Fourth configuration(word transformation: lower case, amount of data used: 50% sampled uniformly at random):

Here, slope = -0.798765745563
R^2 =0.981033608709

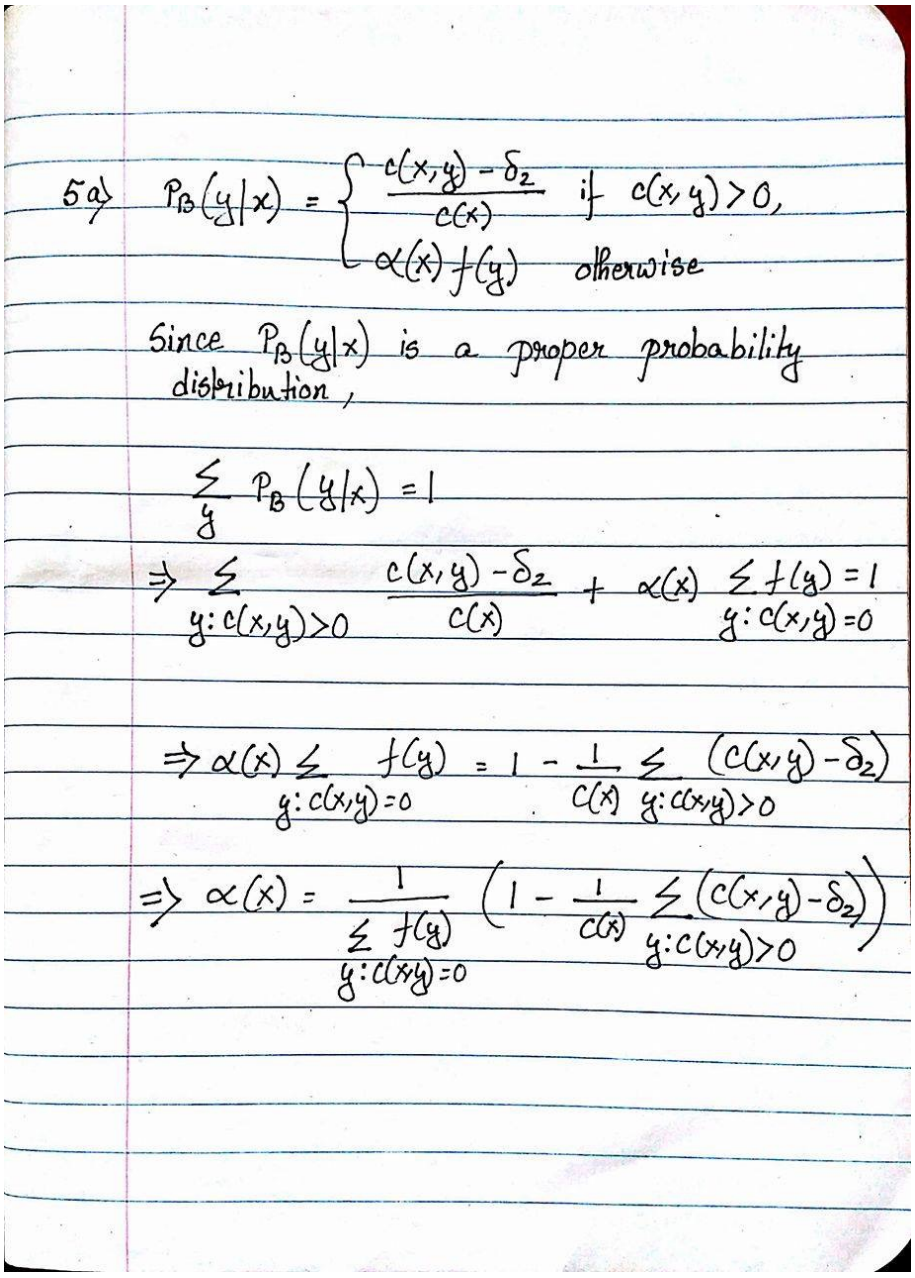
Here the slope is more than in previous case but it's still close to -1 and the value of R^2 is very close to 1 so in this experiment the model fits well.

4.

This paper aims to predict how a particular word fits a context using the association score. They use mutual information between two words to understand their strength of association. Mutual information captures the ratio of the joint probability distribution of the 2 words to the product of their respective marginal distributions. If the mutual information is high, then they probably appear together a large number of times. The paper also shows how to estimate the different probabilities from the corpus. Their major contribution lies in their estimation of the joint probability in which for a word pair (x,y) they calculate the number of times the word “x” appears before “y” (not necessarily just before) in a window size of w. They explain how a short window size captures idioms whereas larger window sizes captures different semantic relationships and conclude that a window size of 5 is a very appropriate one to capture all relationships between the word pairs. This paper provides a statistical theory to measure the association between words which can be calculated very efficiently. Their results are also almost as good as the subjective measures employed by Palermo and Jenkins (1964) .

The paper boasts of a number of findings. Firstly, it shows mathematically why due to limitations in the corpus size, it is difficult to find words which are present in a complementary distribution. Secondly, it shows how phrasal verbs are common with a particular word and goes on to show that the mutual information between such words is mostly comparable. To come up with such findings, they used SInclair’s corpus (British) and the AP corpus (American). They also mention that preprocessing the text before working out the associations gives a clear advantage .

In particular, the Fidditch parser is extremely useful to highlight constraints of interest. In the end they show a few surprising flaws of the Collins dictionary using



5a) $$P_B(y|x) = \begin{cases} \frac{c(x,y) - \delta_2}{c(x)} & \text{if } c(x,y) > 0, \\ \alpha(x) f(y) & \text{otherwise} \end{cases}$$

Since $P_B(y|x)$ is a proper probability distribution,

$$\sum_y P_B(y|x) = 1$$

$$\Rightarrow \sum_{y: c(x,y)>0} \frac{c(x,y) - \delta_2}{c(x)} + \alpha(x) \sum_{y: c(x,y)=0} f(y) = 1$$

$$\Rightarrow \alpha(x) \sum_{y: c(x,y)=0} f(y) = 1 - \frac{1}{c(x)} \sum_{y: c(x,y)>0} (c(x,y) - \delta_2)$$

$$\Rightarrow \alpha(x) = \frac{1}{\sum_{y: c(x,y)=0} f(y)} \left(1 - \frac{1}{c(x)} \sum_{y: c(x,y)>0} (c(x,y) - \delta_2)\right)$$

their technique and mentioned that their method would have been really efficient to remove those flaws.

Overall, this paper has very interesting contributions in this field and they have also shown extensive experiments to corroborate their theoretical analysis of the word association measure.

b) If

$$q_B(y) = \begin{cases} \dfrac{c(y) - \delta_1}{\sum_v c(v)} & \text{if } c(y) > 0, \\ \dfrac{\beta}{V} & \text{otherwise} \end{cases}$$

Since $\sum_y q_B(y) = 1$ [$q_B(y)$ is a proper probability distribution]

$$\Rightarrow \sum_{y: c(y)>0} \frac{c(y) - \delta_1}{\sum_v c(v)} + \frac{\beta}{V} \sum_{y: c(y)=0} 1 = 1$$

$$\Rightarrow \frac{\beta}{V} \sum_{y: c(y)=0} 1 = 1 - \frac{1}{\sum_v c(v)} \sum_{y: c(y)>0} (c(y) - \delta_1)$$

$$\Rightarrow \beta = \left(1 - \frac{1}{\sum_v c(v)} \sum_{y: c(y)>0} (c(y) - \delta_1)\right) \frac{V}{\sum_{y: c(y)=0} 1}$$

5.

$$P_B(y|x) = \begin{cases} \dfrac{c(x,y)-\delta_2}{c(x)} & \text{if } c(x,y)>0 \\ \alpha(x)\dfrac{c(y)-\delta_1}{\sum_v c(v)} & \text{if } c(y)>0,\ c(x,y)=0 \\ \alpha(x)\dfrac{\beta}{V} & \text{if } c(x,y)=0,\ c(y)=0 \end{cases}$$

Since, $\sum_y P_B(y|x) = 1$

$$\Rightarrow \sum_{y:c(x,y)>0} \frac{c(x,y)-\delta_2}{c(x)} + \sum_{\substack{y:c(x,y)=0,\\ c(y)>0}} \alpha(x)\frac{c(y)-\delta_1}{\sum_v c(v)} + \sum_{\substack{y:c(x,y)=0\\ c(y)=0}} \alpha(x)\frac{\beta}{V} = 1$$

$$\Rightarrow \alpha(x)\left[\frac{1}{\sum_v c(v)} \sum_{\substack{y:c(x,y)=0,\\ c(y)>0}} (c(y)-\delta_1) + \sum_{\substack{y:c(x,y)=0\\ c(y)=0}} \frac{\beta}{V}\right] = 1 - \sum_{y:c(x,y)>0} \frac{c(x,y)-\delta_2}{c(x)}$$

$$\alpha(x) = \left(1 - \frac{1}{c(x)} \sum_{y:c(x,y)>0} \frac{(c(x,y)-\delta_2)}{c(x)}\right) \Bigg/ \left[\frac{1}{\sum_v c(v)} \sum_{\substack{y:c(x,y)=0,\\ c(y)>0}} (c(y)-\delta_1) + \sum_{\substack{y:c(x,y)=0\\ c(y)=0}} \frac{\beta}{V}\right]$$

6.

I have written three python codes: Q6.py, Q6_main.py, Q6_main2.py.

Q6.py contains all the required methods.
Q6_main.py is called by the first bash script. It trains a specified model given training and development sets. It then serializes the learned model.
Q6_main2.py is called by the second bash script. It evaluates my saved model on the test data.

I have generated plots for understanding the tuning of parameters using development sets. I have used grid search method to find the best parameters.

The trigram back off model performed best on development sets.

I have considered the 300 least frequent words in training sets to be out of vocabulary words.