Programming Project – 1

Harika Kanakam

Programming project – 1 is developed in python 3.9.12

Task 1: Model Training, Prediction, & Evaluation

To execute the Task 1, the files (*training_data.txt and test_data.txt*) are required which are used to train the models which then is used to the test the model. In this first task, initially train data set is used to train the model and then the test data set is used to evaluate the model. For prediction, three methods are used.

- Using the Maximum Likelihood estimate (MLE)
- Using the MAP estimate (MAP), and
- Using the Predictive Distribution (PD)

Using these 3 methods, unigram learning model is performed and then perplexity of the learned model is calculated on a test data. Perplexity is a standard effectiveness metric in probabilistic language modeling for scoring how well a model predicts a given collection of words (low perplexity values imply good performance).

Note: No smoothing is applied on the data.

The Training contains N (640,000) word in it, in which there are 10000 distinct words. The unigram model is trained using different sizes (N/128, N/64, N/16, N/4, N) of training data.

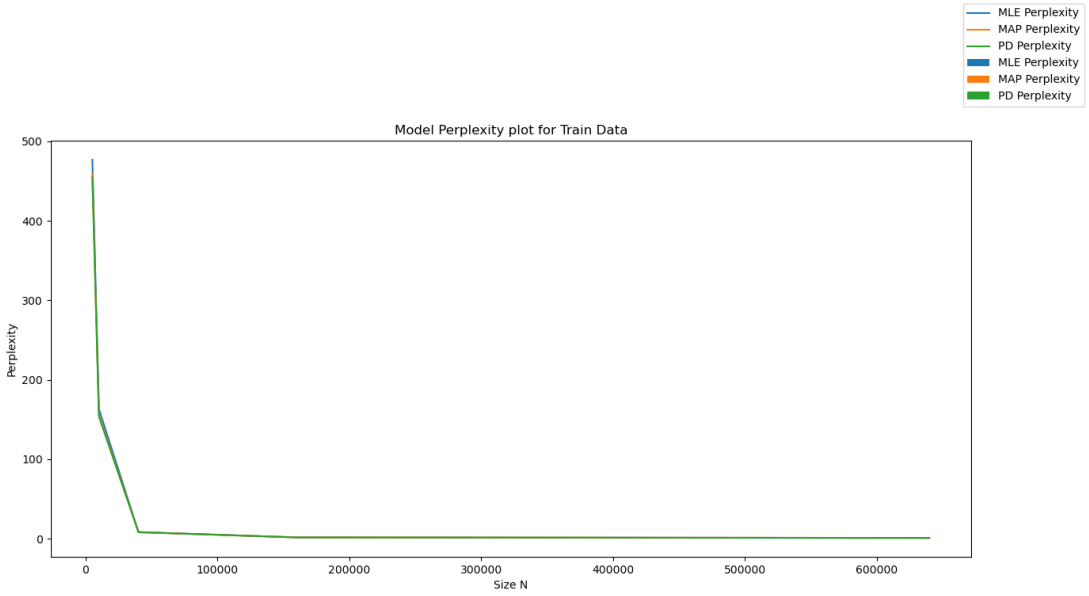
task_1.py code description:

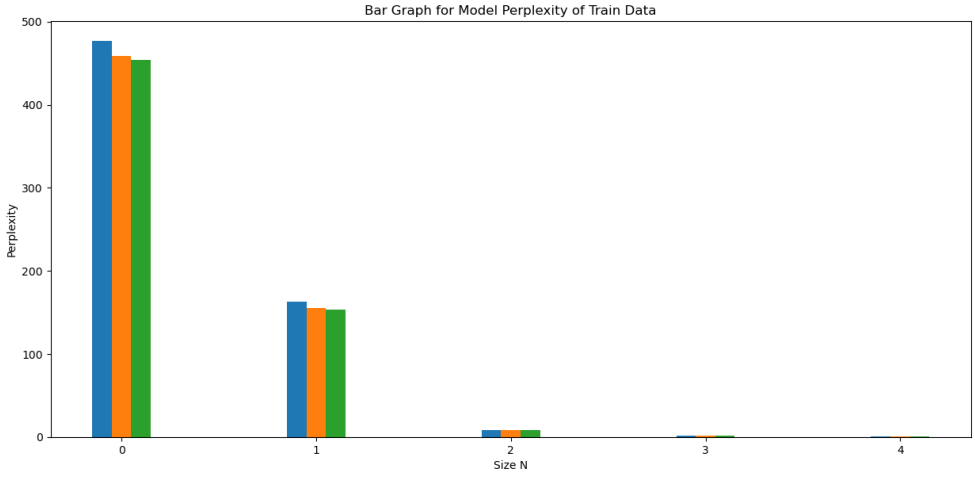
- Initially the 2 files training_data.txt and test_data.txt are read and all the words in the files are stored in file_train_list and file_test_list.
- All the frequencies of distinct words of a train data set are stored in freq_dict.
- The words which are in test data set and not in train data set are stored in freq_dict with a frequency of 0 as these words are not available in train data set.
- Using this freq_dict, all the 3 probabilities – MLE, MAP and PD are for each word in train data is calculated.
- For the train data, the perplexities are calculated which has very low values (This happens as the same data is used to train the model and evaluate the model). The very less values are even occurring because prior is considered when training the model.

Below are the results of the perplexities of the training data set for different sizes of data.

Size	MLE Perplexity	MAP Perplexity	PD Perplexity
N/128	476.8907	459.0254	454.2893
N/64	163.1800	155.7971	153.5703
N/16	8.5697	8.3932	8.3108
N/4	1.7965	1.7926	1.7898
N	1.1581	1.1579	1.1578

The plots for the above result id attached below. As there are 3 models and the plot results are overlapping on each other, Bar graph is also added to the result.



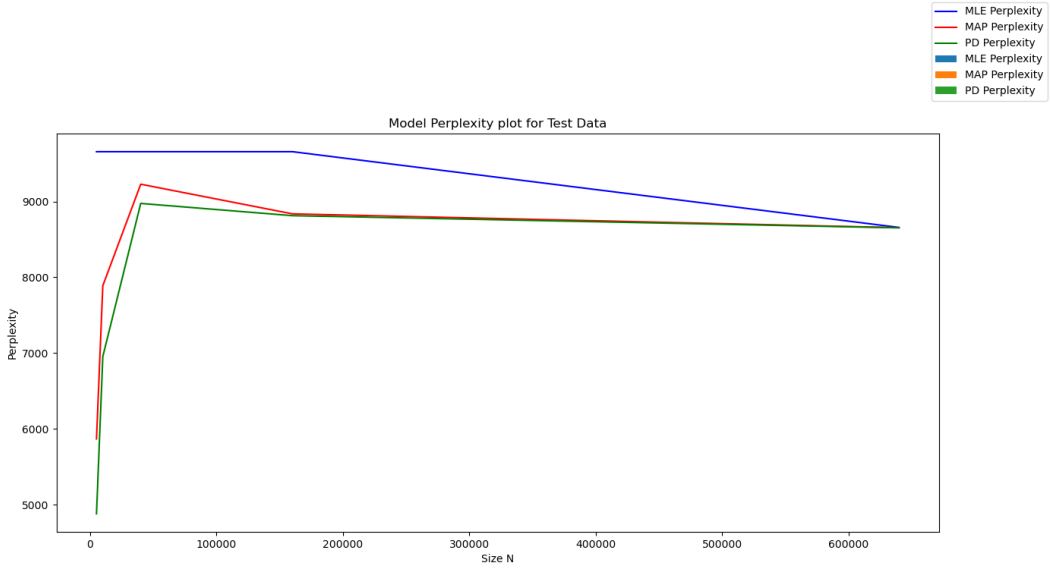


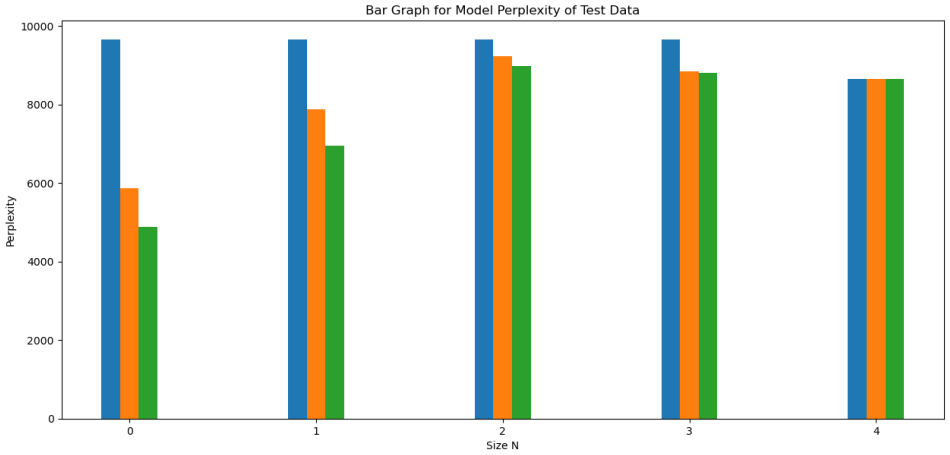
- Using the probabilities that are calculated using train data, perplexities for the test data of these 3 methos are calculated and the results are as below.

Size	MLE Perplexity	MAP Perplexity	PD Perplexity
N/128	Infinity	5867.15	4879.53
N/64	Infinity	7888.97	6954.56
N/16	Infinity	9228.22	8974.53
N/4	Infinity	8837.05	8813.2
N	8657.6230	8654.59	8652.8

- As few MLE perplexities are infinity considered the highest value to plot it in the graph.

The plots for the above result id attached below. As there are 3 models and the plot results are overlapping on each other, Bar graph is also added to the result.





Observation:

- For the MLE method, as there is no prior, the probability of the words that are not in train data becomes zero and log(0) is asked to considered as negative infinity. Hence, the perplexity of that set becomes infinity and is impossible to find the perplexity.
- For the other 2 models as there is prior while calculating the probabilities, the probability doesn't become 0 and hence perplexity is calculated.
- When the training set size increases, it is clearly visible that the plots for the perplexity of the test data using MAP and PD methos is initially increasing and then decreases a little bit and stands in the same range of values as the train set size increase.
- When the train data set size is less, the model mostly depends on the prior and when the train data set size is more, as there is more data, the model doesn't depend on the prior much.
- The obvious shortcoming of the maximum likelihood estimate for a unigram model is that it is possible to calculate the perplexity if and only if all the words of the test set in present in training set. Else, it is impossible to calculate the perplexity.
- For a full training data set, as there is the entire data set to calculate the probabilities of all the unique words, the model doesn't depend on the prior(alph_0 = K*alpha_k) much. So, small changes in alpha_k doesn't effect the perplexity.

Task 2: Model Selection:

To execute the Task 2, the files (*training_data.txt and test_data.txt*) are required which are used to train the models which then is used to the test the model.

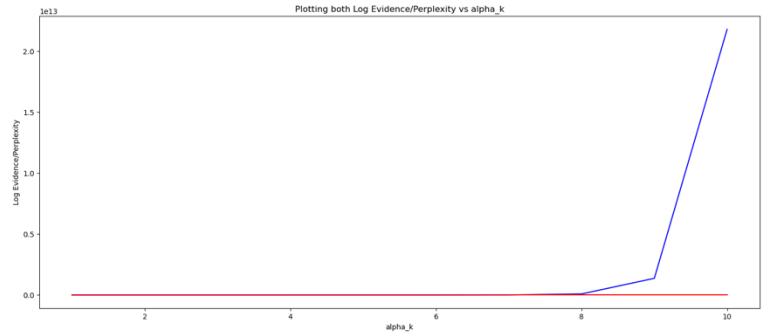
task_2.py code description:

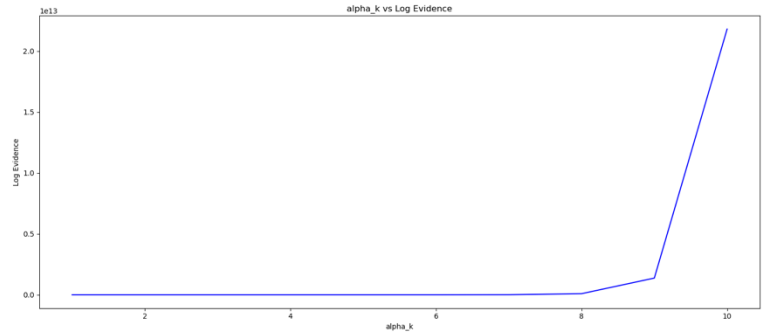
- Initially the 2 files training_data.txt and test_data.txt are read and all the words in the files are stored in file_train_list and file_test_list.
- All the frequencies of distinct words of a train data set are stored in freq_dict.
- The words which are in test data set and not in train data set are stored in freq_dict with a frequency of 0 as these words are not available in train data set.
- Using this freq_dict, the probability(using PD method) for each word in train data is calculated.
- For different values of alpha_k from 1 to 10, the perplexities and the log evidence are calculated for the dataset size of N/128.

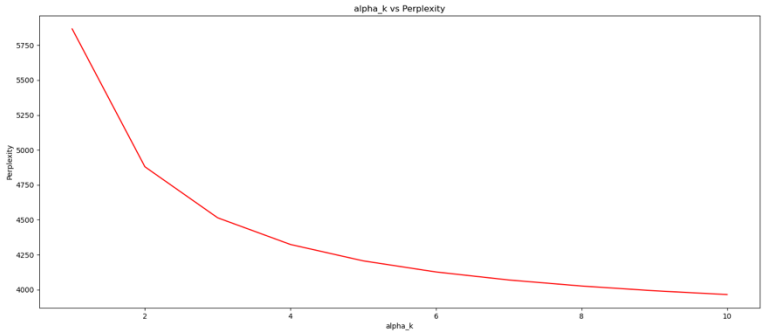
Below are the results for the execution of test2.py

alpha_k	Log Evidence	PD Perplexity
1	-35252.2	5867.15

2	13798.5	4879.53
3	13798.5	4514.18
4	4.27325e+06	4323.22
5	4.52324e+07	4205.81
6	5.2677e+08	4126.34
7	6.70777e+09	4068.98
8	9.25484e+10	4025.63
9	1.37352e+12	3991.73
10	2.1801e+13	3964.49







- Maximizing the evidence function is not suggestable as the values for log evidences increases rapidly with increasing alpha_k value.

Task 3: Author Identification **:**

Initially the probabilities are calculated for all the words that are in dataset pg121.txt.clean file . And the perplexity is calculated for the datasets pg141.txt.clean and pg1400.txt.clean.

The perplexity of the dataset pg141.txt.clean is less than the perplexity of the dataset pg1400.txt.clean. And this is because the author of 121(train dataset) is same as the 141. It is highly possible that the author uses similar words in both books.

Therefore, the model is successful in the classification task.