PROGRAMMING PROJECT 1 REPORT

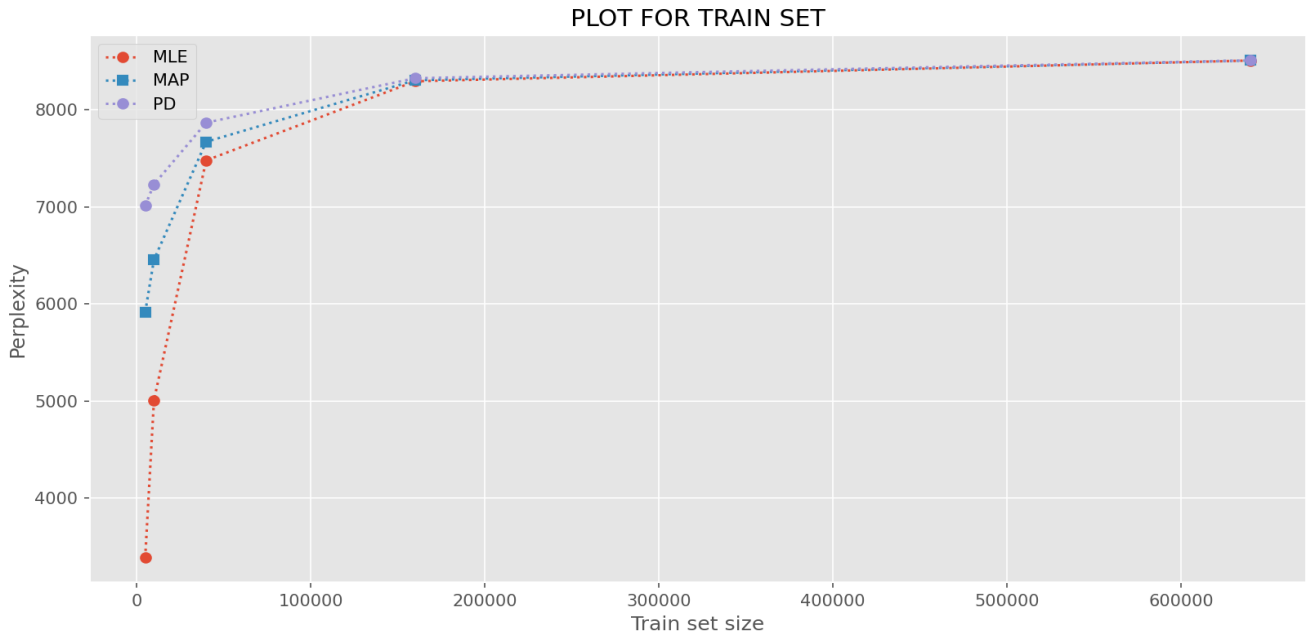
This report contains the analysis and inference of the programming project on the Unigram language model using maximum likelihood, maximum posterior, and predictive distribution estimates. The report is divided into the following sections: (i)implementation details and program flow, (ii)task 1 results and analysis, (iii)task 2 results, (iv) task 3 results, and finally summary in the fourth section.

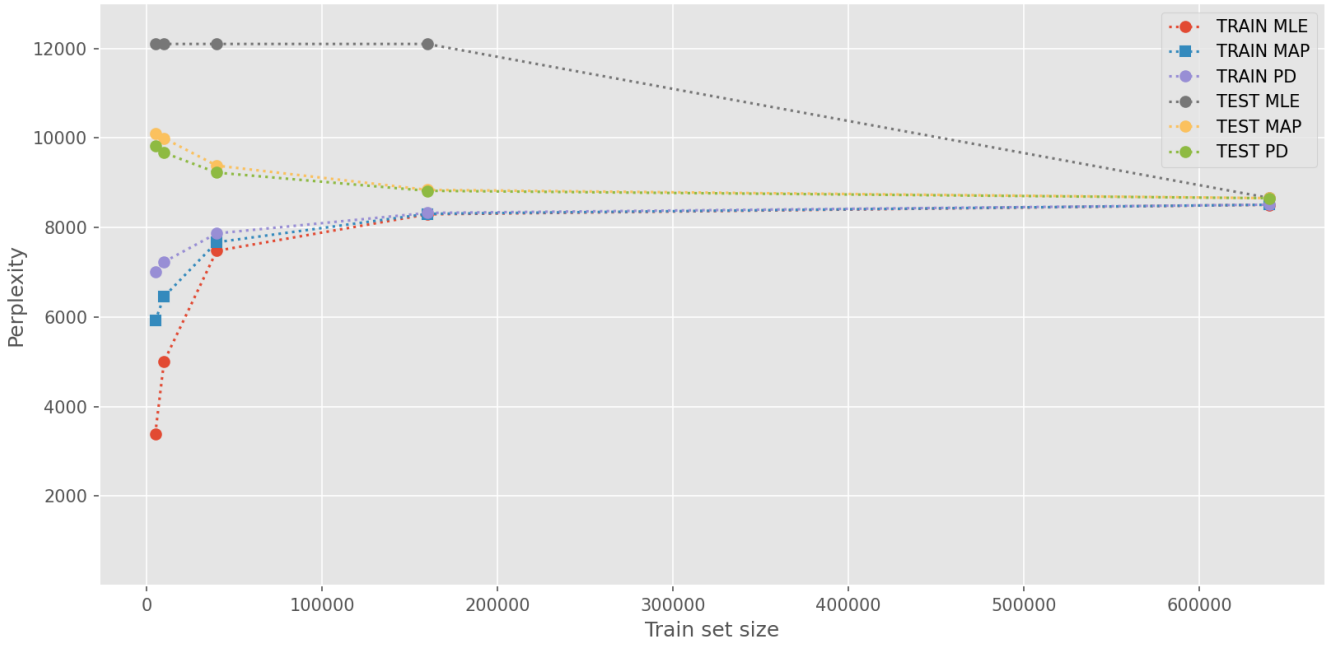
IMPLEMENTATION DETAILS

- The program flow of the implementation is following:
  - It starts with parsing the data from both train and test sets and then calculating the total unique words list
  - Then we iterate over different training set sizes [5000,10000,40000,160000,640000] and calculate the word count for the unique words list and current training set. This is computed by the get_frequency() function.
  - From all the unique words and train word count, we calculate the MLE, MAP, Predictive Distribution (PD), and evidence measures.
  - Then we compute the perplexity for the current training set and the entire testing set. For author identification, we calculate the test perplexity on both of the testing documents.
  - For log evidence we calculate it on the current training set.
  - Finally, we plot the results.

TASK 1

- The task was to plot the test and train perplexities from MLE, MAP, and PD estimates for varying training sizes.
- **RESULTS**





COMBINED PLOT

<Q1> **What happens to the test set perplexities of the different methods with respect to each other as the training set size increases? Please explain why this occurs**

<A1>Perplexity signifies how much the model is "shocked" or perplexed while encountering the words in the test set. If the model has encountered the word before in the training set, then the perplexity would be lower. But if we did training on a limited size of the training set, then the chances of being "perplexed" increase on the test set. So in the testing set, the initial perplexity levels are high, and then it slowly decreases. **But in the training set, we observe that the perplexity increases as we increase the training set size. This is because of overfitting.** When we test the model on which it is trained and the trained data set is small where the model has to deal with less number of words, then due to overfitting the model would perform well, which is having a low bias. But when we test the model on unseen data, it will have a high variance. So that's why there is a disparity in the train and test perplexity.

The training set of size 5000 has a low training perplexity but high testing perplexity. So the model has overfitted to the current training dataset. But as we increase the training size, the overfitting problem is alleviated and the train and test perplexity converge, and there is less disparity. Also, the overfitting problem in MAP and PD is less compared to MLE because we used a prior as a regularizer. Giving an equal amount of belief to each of our words in our vocabulary list would decrease the variance of our model.

```
{'N/128  [TRAIN]':  {'MLE':  3388.2567752661316,  'MAP':
5915.104263875414, 'PD': 7014.415012644447},
'N/64   [TRAIN]':   {'MLE':   5005.38921934077,   'MAP':
6453.994771741911, 'PD': 7230.294305055271},
'N/16   [TRAIN]':   {'MLE':   7478.035656300926,   'MAP':
7669.433287655772, 'PD': 7866.49654406372},
'N/4   [TRAIN]':   {'MLE':   8292.385691235202,   'MAP':
8303.124332826335, 'PD': 8324.24639465811},
'N/1   [TRAIN]':   {'MLE':   8506.433676571029,   'MAP':
8506.965132397268, 'PD': 8508.427803543387}}
```

The test results in dictionary form are the following

```
{'N/128    [TEST]':    {'MLE':    inf,    'MAP':
10098.364924758433,   'PD':   9814.024919830086},
'N/64    [TEST]':    {'MLE':    inf,    'MAP':
9992.362371539855,   'PD':   9668.062580379536},
'N/16    [TEST]':    {'MLE':    inf,    'MAP':
9380.752312209577, 'PD': 9224.511912782404}, 'N/4
```

```
[TEST]': {'MLE': inf, 'MAP': 8839.546029240897,
'PD': 8817.90483983308}, 'N/1 [TEST]': {'MLE':
8657.623041670242, 'MAP': 8654.59009099356, 'PD':
8652.803792546738}}
```

Here for first three cases, the MLE is infinite and it has been capped while plotting. The MAP and PD values are finite as we have set a prior belief and assigned the frequency of each word to be 1 instead of 0 as in MLE. For MAP and PD, perplexity decreases as the training set size increases. So increasing the training data, increases the accuracy of our model.

<Q2>**What is the obvious shortcoming of the maximum likelihood estimate for a unigram model? How do the other two approaches address this issue?**

The obvious shortcoming of MLE is that we don't have a prior like a Dirichlet distribution. But for MAP and PD we have a prior where we believe that each word should appear at least once in the training set as α`=2. But MLE has a frequency of 0, so the test perplexity goes to infinite, which does not help in analyzing the effect of various training sizes. Also, the MAP and PD estimates update their beliefs based on the training set and as the training size increases, their perplexity is more efficient compared to MLE. The overfitting problem and high variance is more pronounced in MLE, as it only maximizes its estimates based on the dataset. So if the dataset is

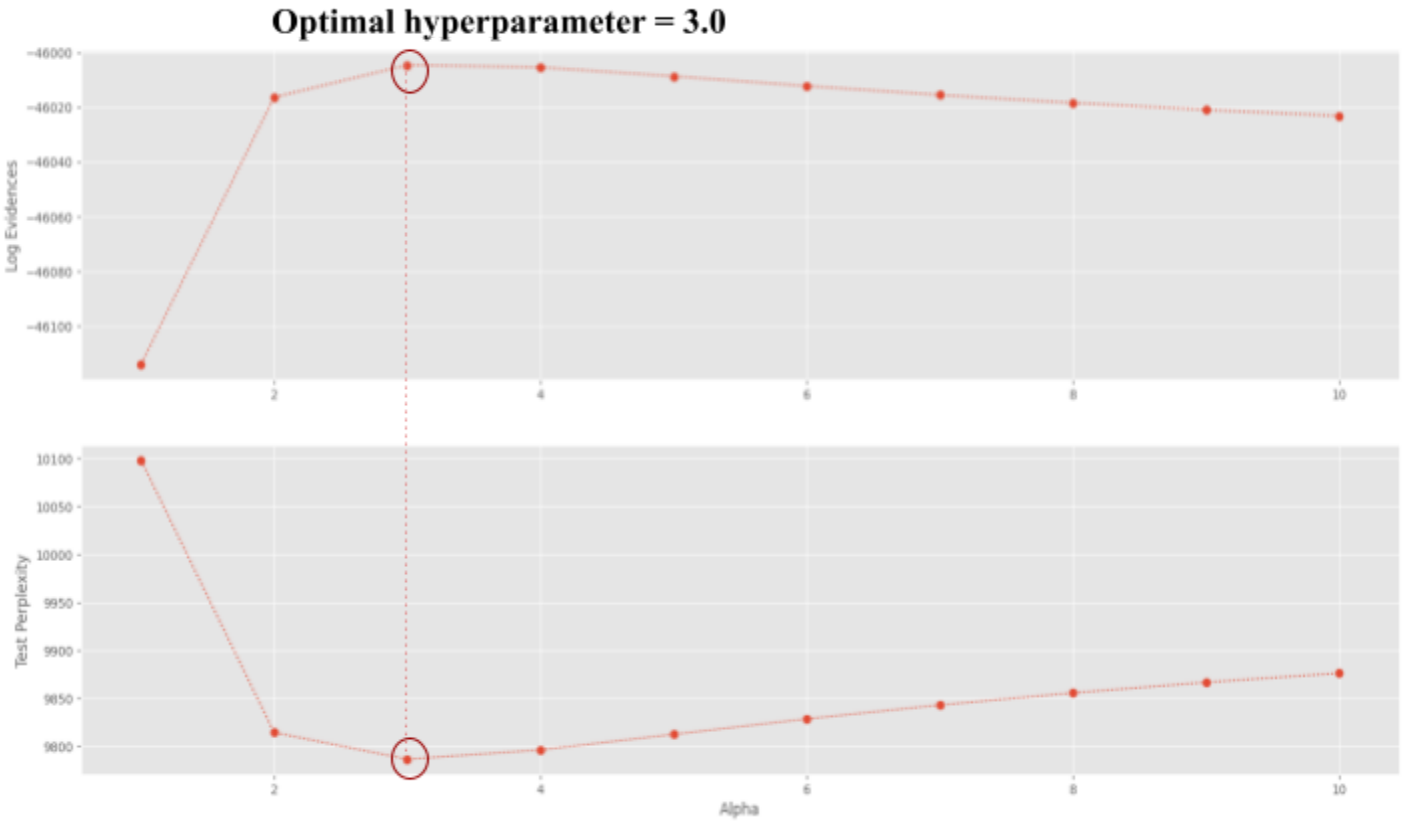
small and skewed, we may overfit to our training data. But MAP and PD have a prior as a regularizer to overcome the overfitting problem.

<Q3>**For the full training set, how sensitive do you think the test set perplexity will be to small changes in α`? Why?**

The alpha value, which is a hyperparameter in the Dirichlet distribution, signifies the prior. The pseudo count of **α` is α`-1.** So if we have a prior of 2, the pseudo count is 1. Thus we believe each word will appear at least once in the training set. But if we increase the prior, then we are fully committing to the prior, like our belief about the frequency of each word is high. So it may not be optimal as we should always pick a less committed prior and then update the prior belief using the training data. This will provide us with more flexibility. Also, a strong prior will be hard to change or update. So in the test set perplexity, we will see that the perplexity will start to decrease initially when we start from 1, then after a certain point (in this task it is 3), it will start to increase which tells us that having a highly committed prior is not suitable. Also, a very low prior is not suitable as in a small dataset, some of the words will have high frequency thus the model might overfit on these words. So having a prior in the middle ground can help us to regularize as well.

TASK 2

- This task deals with hyperparameter optimization of the Dirichlet distribution and plotting the log evidence and test set perplexity using Predictive distribution for a given training size of 5000.



- The Dirichlet distribution has a hyperparameter α` which assigns pseudo counts to each word in our vocabulary, that is prior. So if α` is N, then the pseudo count is N-1. So if we have a lower α`, then the prior belief for the occurrence of each word is low but if we have a higher α` then the prior belief is high. So we have a less committed prior in the former and in highly committed prior in the laters.
- So the plot suggests that we need to find a goldilocks zone between these two commitments, that is not too high or not too

low. A highly committed prior will assign a high frequency to each word and probably for a smaller dataset, it will override the probability estimates that we learned from the data. Thus it will not update the model, hence hard to update. But our main aim is to update the prior so as to fit the training data.

- With a less committed prior, maybe we will overfit the training dataset and our model will be biased to frequent words and not perform well on unseen datasets. Thus it may have a low bias (low perplexity on training data) but high variance (high perplexity on testing data). So we need to level the playing field anticipating that the model may overfit some data.
- So α`=3.0 turns out to be the right value for the hyperparameter to have the Dirichlet distribution as it gives the highest log evidence value and lowest test perplexity value.

Is maximizing the evidence function a good method for model selection on this dataset?

- Hyperparameter optimization is done while training the data, because we don't have the test perplexity results while training. But log-evidence is a good surrogate function for the test perplexity as it results in the same inferences that we will get after testing the dataset. So while training, log evidence values can help us to pick an optimal hyperparameter from the training set itself. This would help us to define a prior that will give us a low perplexity value in the test set.

TASK 3

The task was to compare the perplexity between two documents written by different authors after training a model using predictive

distribution from one of the models. The training used was N/128, where N is the total number of words in the document.

RESULTS

- So the model was trained on pg121 text and it was tested on pg141 and pg1400. The former was written by the same author as the training set (pg121) and the latter was written by a different author. So we should expect the perplexity of the document belonging to the same author to be low and another one to be high.
- So in the above plot, we can see that pg141 has a lower perplexity compared to pg1400. This reinforces the idea that the model is less perplexed by testing over the document from the same author but it had higher perplexity for another author.

- Authors writing style and choice of words tend to be similar, which we have shown empirically in our testing by finding a higher perplexity in the test set belonging to a different author.