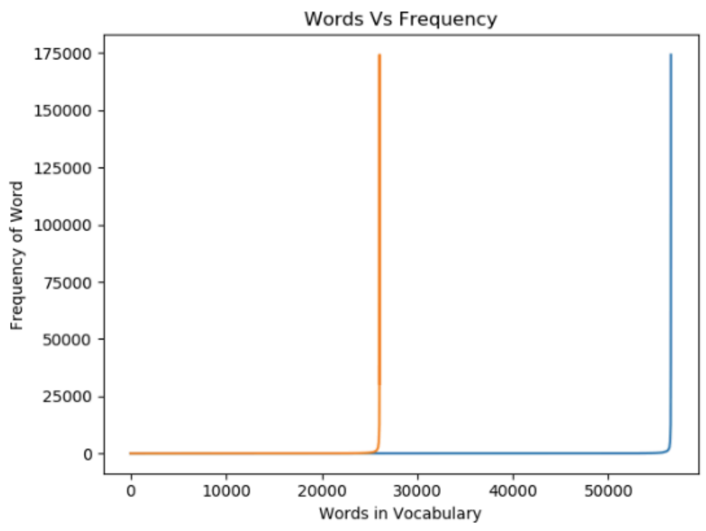
NLP Homework 1
Brandon Peck
bjp9pq

2)
1)
I made every word lower case and stripped the punctuation which changed the total word count in the train set from 72,163 to 56,548. I determined that there are 30,488 words which are only counted once and that is equal to 53.9% of the total number of words. I mapped those words to UNK to reduce the number of features. The graph below shows the change in word frequency distribution after the single frequency words are removed.



Key: The blue line shows the word frequency including single frequency words, and the orange line excludes those words.
The resulting feature vector, f(x,y), is one dimensional encoded as

$$\boldsymbol{f}(\boldsymbol{x}, y = K) = [\underbrace{0; 0; \ldots; 0}_{(K-1)\times V}; \boldsymbol{x}], \qquad [2.5]$$

(JE)

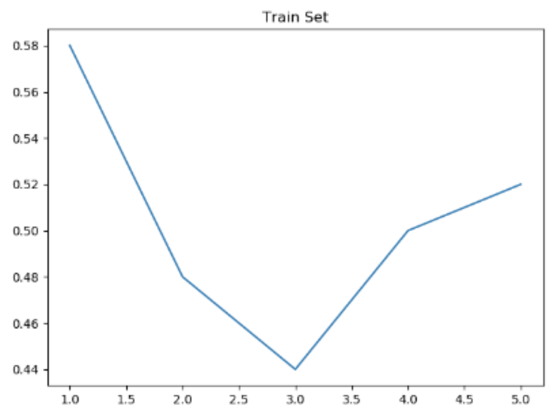
Where V is the size of the vocabulary. The **feature set size is V x range(Y) or 4254 * 2= 8507**. This is because since there are 4254 words including the UNK word and two features per word. The resulting feature vector is encoded similar to this:
Input x, y vectors:
[‘The’, ‘staff’, ‘was’, ‘rude.’] [0]
[‘The’, ‘staff’, ‘was’, ‘friendly.’] [1]

Output feature vector:

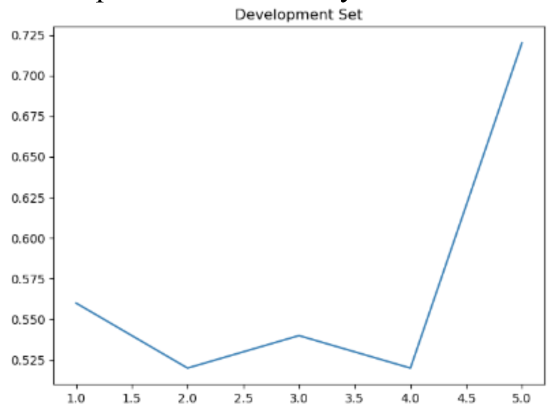
the+	staff+	was+	friendly+	rude+	UNK+	the-	staff-	was-	friendly-	rude-	UNK-
1	1	1	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	0	1	0

With the real data the least frequent words map to UNK which I didn't show in the example above.
2)
My computer could not handle using all of the data so I applied the model to a sample of n=5,000

Training Set Accuracy Plot:



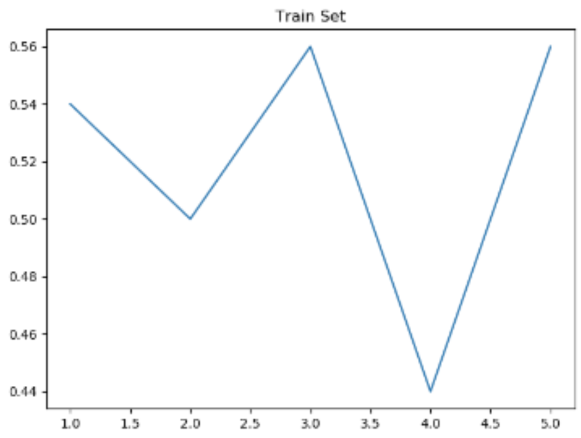
*Y axis is accuracy X axis is epoch number

Development Set Accuracy Plot:


*Y axis is accuracy X axis is epoch number

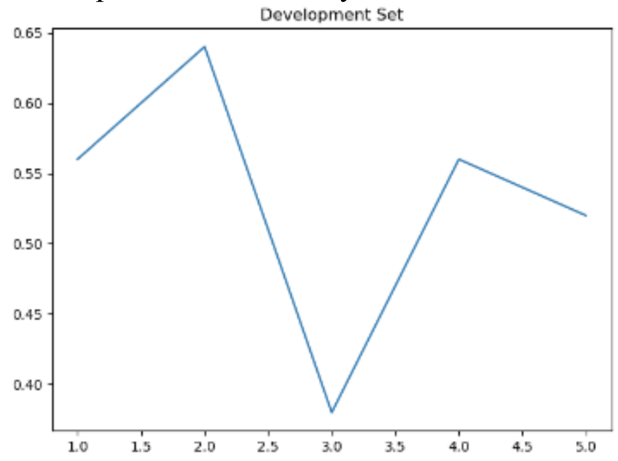
3)

Training Set Accuracy Plot:



*Y axis is accuracy X axis is epoch number

Development Set Accuracy Plot:



*Y axis is accuracy X axis is epoch number

3)
1)
To find the accuracy, I averaged the scores across a 3-Fold cross validated split.
The feature set size is: 38,911 words
The classification accuracy on the training set is: 87.473%
The classification accuracy on the development set is: 86.329%

2)
The feature set size is: 475,575 grams
The classification accuracy on the training set is: 89.717%
The classification accuracy on the development set is: 86.94%

3)

Lambda	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10	100	1000
Accuracy Training	84.6%	84.7%	84.875	85.15%	85.57%	85.85%	85.91%	85.5%
Accuracy Development	82.42%	83.28%	83.98%	84.52%	85.55%	83.88%	83.52%	78.7%

From the above it looks like there's a maximization between 100 and 1000

Trying to narrow the lambda range:

Lambda	100	300	500	700	1000
Accuracy Training	86.28%	85.53%	84.98%	84.55%	84.09%
Accuracy Development	83.52%	82.49%	81.77%	81.22%	80.716%

The optimal lambda is likely in the order of 10^1 to 10^2.

4)

L1 regularization sparsity explanation:
We know that the loss function is a function of the input x applied by theta to produce a predicted output. The loss function between the true and predicted outputs is used to penalize the values of theta and move it towards a minimal predicted value. The contour plot is a visualization of thetas on various dimensions plotted against each other and how they respond to the loss function. The L1 regularization adds a boundary on the sum of the theta vectors.
The regularization term when added to the loss function produces a new minimizing area between the minimized loss function and edge of the regularization boundary. In a sparse dataset we expect specific theta vectors to have more weight over others so we want to bias toward a theta. Because L1 regularization has a maximal radius when one theta dimension is maximal when compared to another it achieves better performance with sparsity than L2, which has equal radius across both dimensions of theta.

Lambda	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10	100	1000
Accuracy Training	83.68%	84.92%	85.01%	85.57%	86.73%	79.61%	63.47%	49.74%
Accuracy Development	81.1%	83.96%	83.18%	83.32%	82.84%	71.8%	49.28%	49.28%

The lambda range is smaller here and is likely in the order of 10^{-3} and 10^{-2}

5)
I changed the CountVectorizers ngram_range to (1,2) effectively doubling the feature set by adding Bi-grams. As discussed in lecture and from experimenting above we expect a larger and richer feature set to increase the accuracy of the model. This is because a richer feature set describes an input x more succinctly. However more features using Bi-grams increase the potential for overfitting because you can over bias specific features, as they can be counted twice

if specific phrases almost always come together. Based on the expected sparsity of the dataset, as there are many words in the vocabulary that are not used in each document and I doubled the feature vector size, I chose to use L1 regularization. I tried using the stochastic average gradient solver since we discussed in class that this modifies the prediction vector by some learning rate in a direction towards minimizing the error of the loss function. The model printed a convergence warning, stating that the minimal gradient value was not reached however, so I used the liblinear solver.

The best model accuracy I found was using L1 regularization, ngram_range (1,2), and liblinear solver:
Training set accuracy: 89.82%
Development set accuracy: 86.67%