ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Text Analytics
Exercise 2 Report

Papastamos Konstantinos – DS3617011

Kanellis Georgios – DS3617006

Tsolas Leonidas– DS3617017

Exercise

The exercise given is described below:

Develop a text classifier for a kind of texts of your choice (e.g., e-mail messages, tweets, customer reviews) and at least two classes (e.g., spam/ham, positive/negative/neutral).

You may use Boolean, TF, or TF-IDF features corresponding to words or n-grams, to which you can also add other features (e.g., length of the text). You may apply any feature selection (or dimensionality reduction) method you consider appropriate. You may also want to try using centroids of pre-trained word embeddings.

You can write your own code to produce feature vectors, perform feature selection (or dimensionality reduction) and train the classifier (e.g., 2 For e-mail spam filtering, you may want to use the Ling-Spam or Enron-Spam datasets (available from http://nlp.cs.aueb.gr/software.html). For tweets, you may want to use datasets from http://alt.qcri.org/semeval2016/task4/. For customer reviews, you may want to use datasets from

http://alt.qcri.org/semeval2016/task5/.

Consult the instructor for further details. Pre-trained word embeddings are available, for example, from

http://nlp.stanford.edu/projects/glove/.

See also word2vec (https://code.google.com/archive/p/word2vec/). (-1,1): -1 (1,1): -1 (-1,-1): 1 (1,-1): 1 1 -1 0 x1 x1x2 (+1) (+1) 1 0 (-1) -1 (-1) x1 x2 1 -1 using SGD, in the case of logistic regression), or you can use existing implementations and software libraries.

You should experiment with at least logistic regression, and optionally other learning algorithms (e.g., Naive Bayes, k-NN, SVM). Draw learning curves (slides 58, 61) with appropriate measures (e.g., accuracy, F1) and precision-recall curves (slide 26). Include experimental results of appropriate baselines (e.g., classifiers that always assign the most frequent class). Make sure that you use separate training and test data. Tune the feature set and hyper-parameters (e.g., regularization weight λ) on a held-out part of the training data or using a cross-validation (slide 28) on the training data. Document clearly in a short report (max. 5 pages) how your system works (e.g., what algorithms it uses, examples of input/output) and its experimental results (e.g., learning curves, precision-recall curves).

For our implementation we used python and the nltk package. Below all steps taken will be described in detail along with the necessary result screenshots.

The full code used for the exercise can be found here.

1. Data Loading and Preprocessing

First the file of kaggle's imdb reviews dataset was downloaded as found here. Only the "pos"and "neg" folders were kept. After the files were ready, the script iterated over the files, loaded the raw text and used the Clean_Text() function on each individual sentence. The mentioned function does the following:

1. Turns everything to lower case.
2. Removes punctuation
3. Reduces all whitespace down to one.

The result of the loading and preprocessing steps is two lists, "data" which is a list of strings containing the sentences of all files and "labels" which is a list containing 1s and 0s depending on whether the respective sentence in "data" was read from the pos (positive reviews) or neg (negative reviews) folders.

2. Feature Extraction

On this step, the TfidfVectorizer function was used from the sklearn package to create unigram and bigram Tf-Idf features. Two different instances were created for that purpose:

Unigram_vectorizer

The unigram vectorizer created a sparse matrix of Tf-Idf features. Just for unigrams we decided to remove stopwords in order to reduce dimensionality and also because we made the assumption that stopwords do not add information on the sentiment of a sentence when working with unigrams. Finally unigrams with document frequency less than 10 were ignored.

Bigram_vectorizer

The Bigram vectorizer created a sparse matrix of Bigram counts. Bigrams with document frequency less than 10 were ignored.

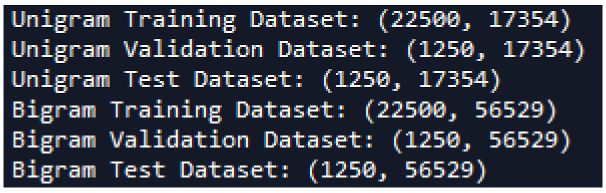
3. Data Shaping and Preprocessing

3.1. Set Formation

The next step was to randomly split the corpus into training, validation and test sets. The splitting was done two times, one for unigrams and one for bigrams resulting in 6 different sets. The splitting was done using sklearn's train_test_split function.

A 90 - 5 - 5 split strategy was used to form the training, validation and test sets.

The resulting shapes are the following:

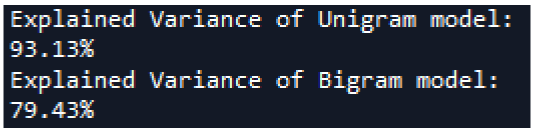


```
Unigram Training Dataset: (22500, 17354)
Unigram Validation Dataset: (1250, 17354)
Unigram Test Dataset: (1250, 17354)
Bigram Training Dataset: (22500, 56529)
Bigram Validation Dataset: (1250, 56529)
Bigram Test Dataset: (1250, 56529)
```

3.2. Dimensionality Reduction

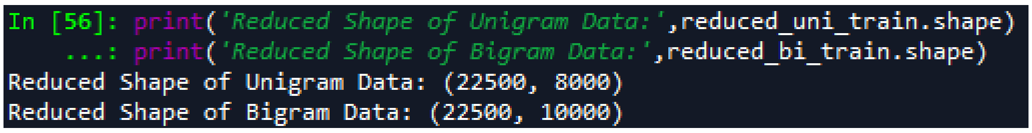
However after inspecting the resulting datasets, the number of created features is obviously too large. So we decided to use truncated SVD in order to reduce the dimensions as much as possible.

We used the TruncatedSVD module from the sklearn.decomposition package and performed SVD on both the unigram and bigram training sets. For the unigram training set we kept 8000 components while for the bigram set we kept 10000, since the bigram set had initially many more features. The choice was made after several tries and by consulting the explained variance ratio for different number of components. The resulting sets have an explained variance ratio as seen below:



```
Explained Variance of Unigram model:
93.13%
Explained Variance of Bigram model:
79.43%
```

So the resulting dimensions after using Truncated SVD are:



```
In [56]: print('Reduced Shape of Unigram Data:',reduced_uni_train.shape)
    ...: print('Reduced Shape of Bigram Data:',reduced_bi_train.shape)
Reduced Shape of Unigram Data: (22500, 8000)
Reduced Shape of Bigram Data: (22500, 10000)
```

We decided to not try to keep more components in order to prevent models from overfitting.

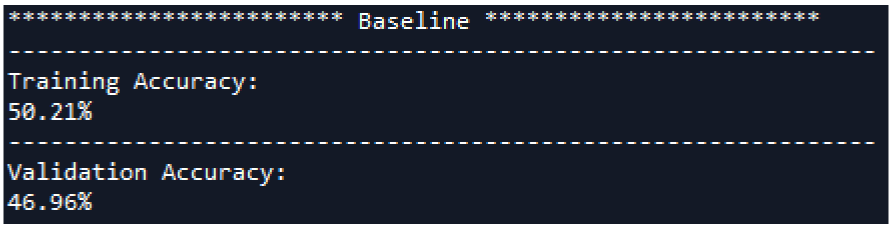
4. Modelling

On the next step we used the sklearn package to train different classifiers and evaluate the results. First we created a baseline model to use as a frame of reference and then we experimented with 2 different models, Naïve Bayes and Logistic Regression. The best model will be tuned in order to improve performance further.

4.1. Baseline

The baseline-dummy classifier generates always the class 0, since it is the most frequent class on the training set, although the dataset is balanced, so the class frequency is almost the same for both classes.

Thus the baseline scores are seen below:



```
************************ Baseline ************************
--------------------------------------------------------------
Training Accuracy:
50.21%
--------------------------------------------------------------
Validation Accuracy:
46.96%
```

As expected for a balanced dataset, a model that generates always the same class will get almost 50% accuracy.

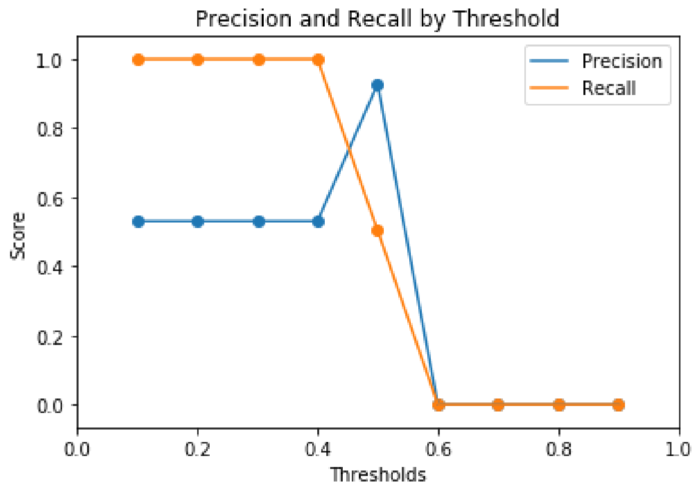
4.2. Naïve Bayes

Training

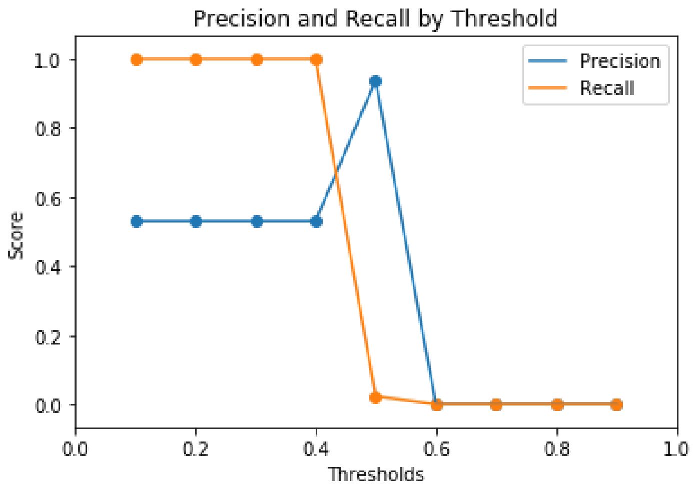
The first model we experimented with is Naïve Bayes. We trained the Multinomial Naïve Bayes model since its more suitable for classification with discrete features. The model was trained on both the unigram and bigram sets. In order for the model to work, we adjusted the training data by adding a constant equal to the absolute value of the minimum X data value, in order to move all training points above the X axis (since the multinomial NB model only accepts non-negative values as input).

Tuning

In both models we tuned the alpha (smoothing) parameter as well as the prediction threshold. The alpha parameter didn't seem to affect the final scores so we used the default value for our experiments. However the prediction threshold was very sensitive to different values. Below are the precision and recall curves for different thresholds for both Unigram and Bigram data:

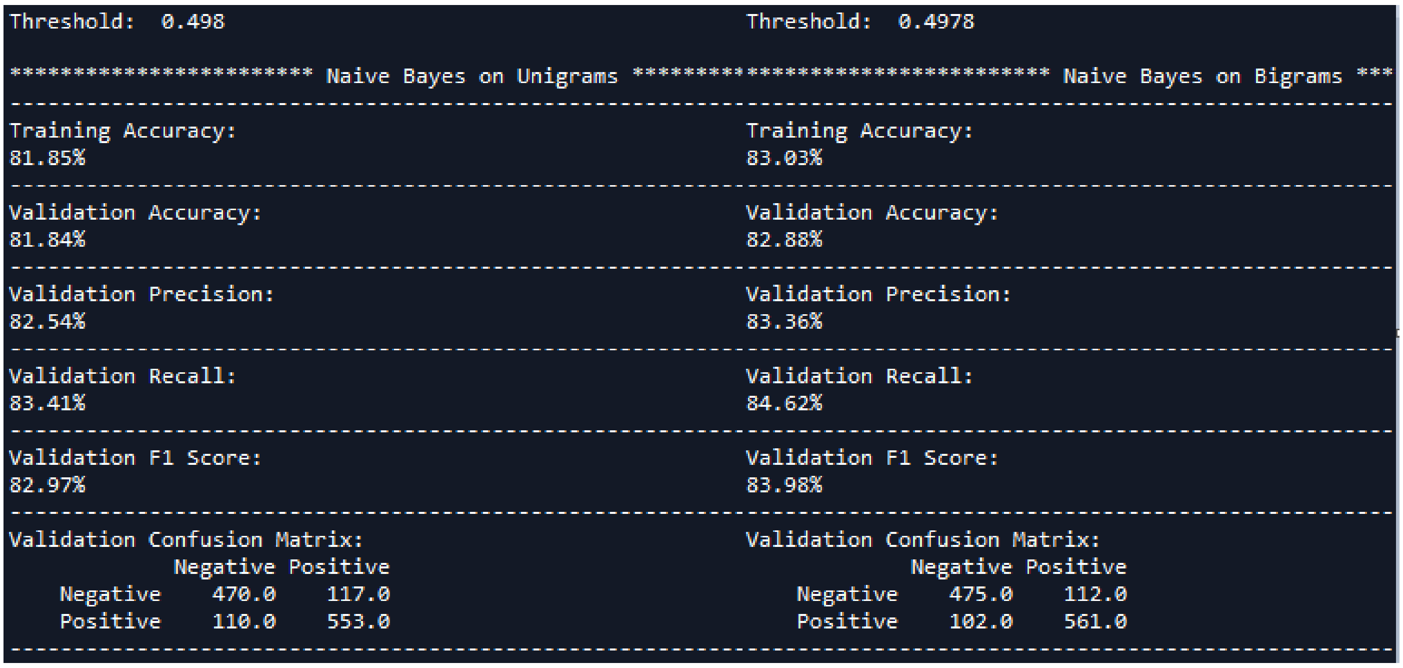


Unigrams



Bigrams

It's clear that the optimal threshold is a bit below 0.5. After careful tuning we estimated that the optimal value is approximately 0.498 for the Unigram dataset and 0.4978 for the Bigram dataset. The analytical tuning results can be found on the files "Naive Bayes Threshold Tuning on Bigrams.txt" and "Naive Bayes Threshold Tuning on Unigrams.txt" on the Naïve Bayes folder in here. With the current hyperparameter values the Naïve Bayes model yields the following results:



```
Threshold:  0.498                                          Threshold:  0.4978

************************ Naive Bayes on Unigrams ********************************* Naive Bayes on Bigrams ***
---------------------------------------------------------------------------------------------------------------
Training Accuracy:                                         Training Accuracy:
81.85%                                                     83.03%
---------------------------------------------------------------------------------------------------------------
Validation Accuracy:                                       Validation Accuracy:
81.84%                                                     82.88%
---------------------------------------------------------------------------------------------------------------
Validation Precision:                                      Validation Precision:
82.54%                                                     83.36%
---------------------------------------------------------------------------------------------------------------
Validation Recall:                                         Validation Recall:
83.41%                                                     84.62%
---------------------------------------------------------------------------------------------------------------
Validation F1 Score:                                       Validation F1 Score:
82.97%                                                     83.98%
---------------------------------------------------------------------------------------------------------------
Validation Confusion Matrix:                               Validation Confusion Matrix:
             Negative Positive                                          Negative Positive
    Negative    470.0    117.0                                 Negative    475.0    112.0
    Positive    110.0    553.0                                 Positive    102.0    561.0
---------------------------------------------------------------------------------------------------------------
```

Conclusions:

Naïve Bayes seems to be an improvement from the baseline model. On the both datasets there is a great boost on the scores, with the bigram model performing slightly better, and the difference between the training and validation scores is small, indicating that the classifier is generalizing well. Even though the scores don't seem to reach very high values, the model seems to be stable and reliable.

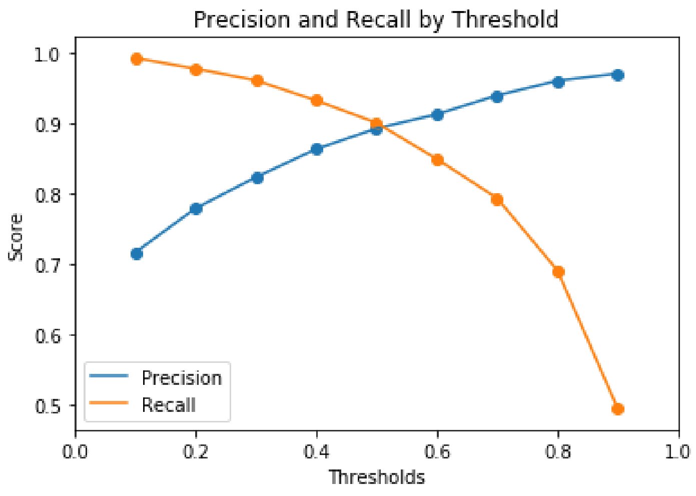
4.3. Logistic Regression

Regularization Tuning

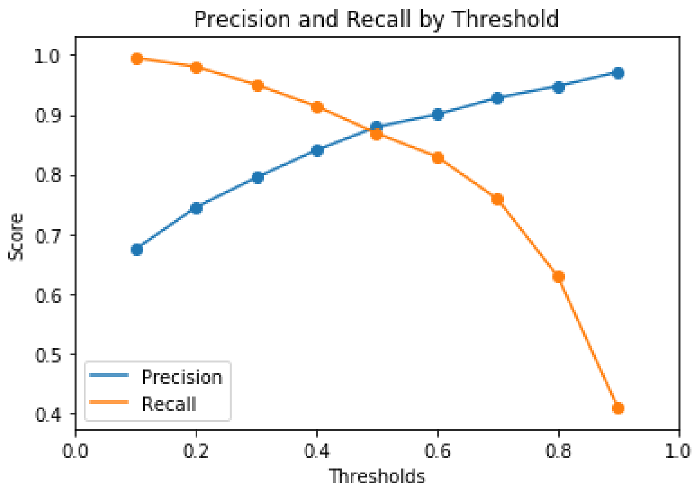
Next we trained a logistic regression classifier. For both Unigrams and Bigrams, we tuned the regularization as well as the penalty parameters for different values and concluded on using l1 penalty with C = 0.9 since it resulted on the highest validation accuracy without overfitting (validation accuracy was close to training accuracy). The tuning results can be found in the Logistic Regression folder in here.

Threshold Tuning

We also tried to tune the prediction threshold but as seen below, the optimal threshold seems to be the default 0.5 so that what we will use for the final model.



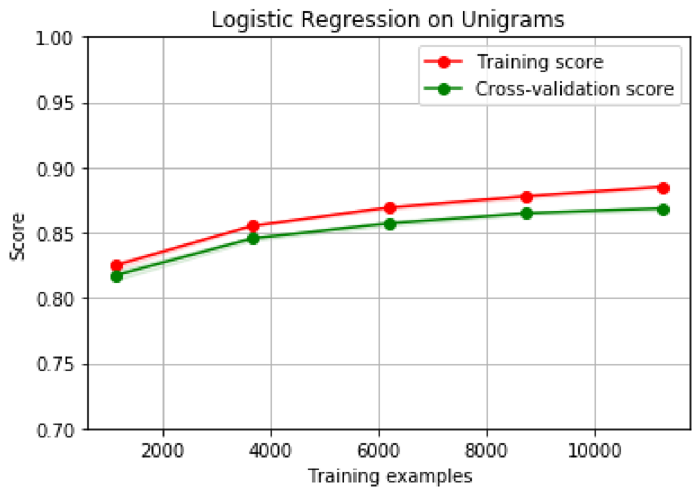
For Unigrams

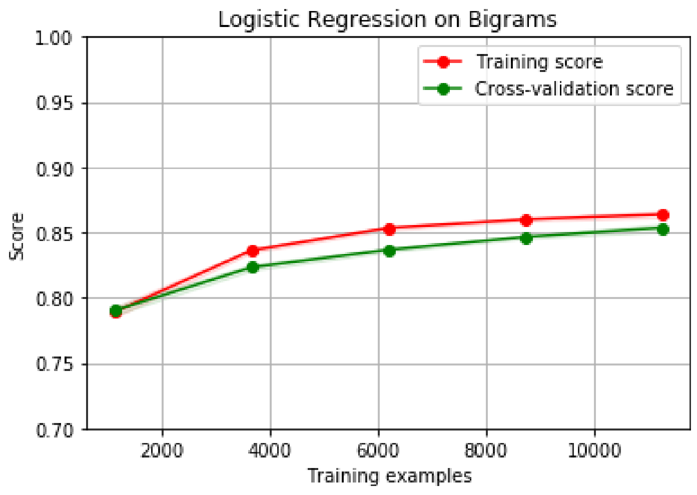


For Bigrams

Learning Curves

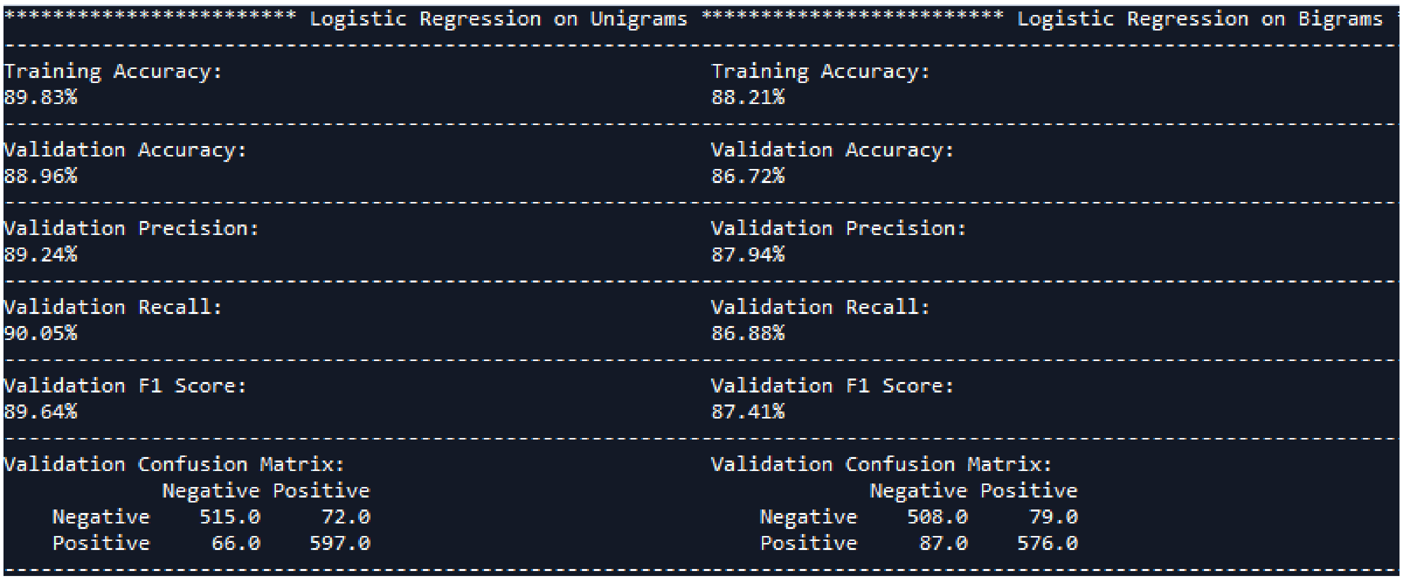
In order to ensure that the training is smooth we also consulted the learning curves.





After tuning of the regularization, the training seems to be without issues. The training and validation scores are increasing with training size, which means that we do not provide unnecessary data, and they do not have a significant distance indicating that the model is generalizing well.

The final results can be seen below:



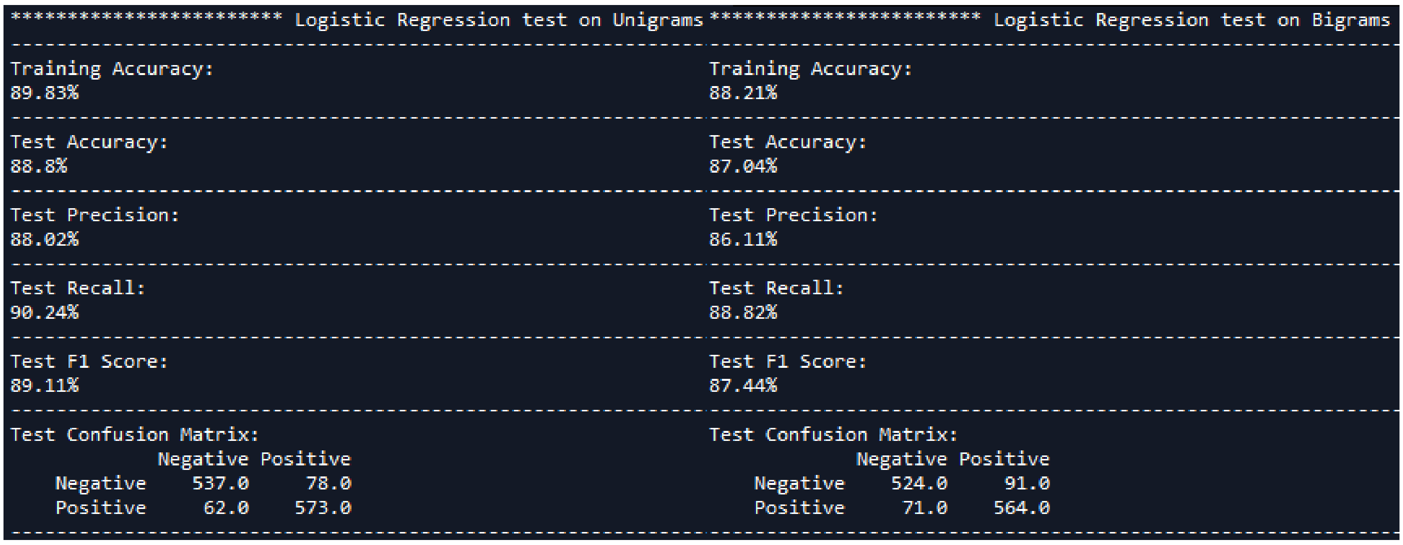
```
************************ Logistic Regression on Unigrams ************************ Logistic Regression on Bigrams
----------------------------------------------------------------------------------------------------------------
Training Accuracy:                                        Training Accuracy:
89.83%                                                    88.21%
----------------------------------------------------------------------------------------------------------------
Validation Accuracy:                                      Validation Accuracy:
88.96%                                                    86.72%
----------------------------------------------------------------------------------------------------------------
Validation Precision:                                     Validation Precision:
89.24%                                                    87.94%
----------------------------------------------------------------------------------------------------------------
Validation Recall:                                        Validation Recall:
90.05%                                                    86.88%
----------------------------------------------------------------------------------------------------------------
Validation F1 Score:                                      Validation F1 Score:
89.64%                                                    87.41%
----------------------------------------------------------------------------------------------------------------
Validation Confusion Matrix:                              Validation Confusion Matrix:
             Negative Positive                                         Negative Positive
    Negative    515.0     72.0                                Negative    508.0     79.0
    Positive     66.0    597.0                                Positive     87.0    576.0
----------------------------------------------------------------------------------------------------------------
```

Conclusions:

Logistic Regression results in even greater predictions than naïve bayes, and after tuning is also stable without overfitting on the training data. The model performs slightly better on the Unigram dataset.

Testing

Since Logistic Regression performed better than Naïve Bayes, that’s the model we will use to evaluate the test set. The test results can be seen below:



```
************************ Logistic Regression test on Unigrams ************************ Logistic Regression test on Bigrams
--------------------------------------------------------------------------------------------------------------------------
Training Accuracy:                                            Training Accuracy:
89.83%                                                        88.21%
--------------------------------------------------------------------------------------------------------------------------
Test Accuracy:                                                Test Accuracy:
88.8%                                                         87.04%
--------------------------------------------------------------------------------------------------------------------------
Test Precision:                                               Test Precision:
88.02%                                                        86.11%
--------------------------------------------------------------------------------------------------------------------------
Test Recall:                                                  Test Recall:
90.24%                                                        88.82%
--------------------------------------------------------------------------------------------------------------------------
Test F1 Score:                                                Test F1 Score:
89.11%                                                        87.44%
--------------------------------------------------------------------------------------------------------------------------
Test Confusion Matrix:                                        Test Confusion Matrix:
             Negative Positive                                             Negative Positive
    Negative    537.0     78.0                                    Negative    524.0     91.0
    Positive     62.0    573.0                                    Positive     71.0    564.0
--------------------------------------------------------------------------------------------------------------------------
```