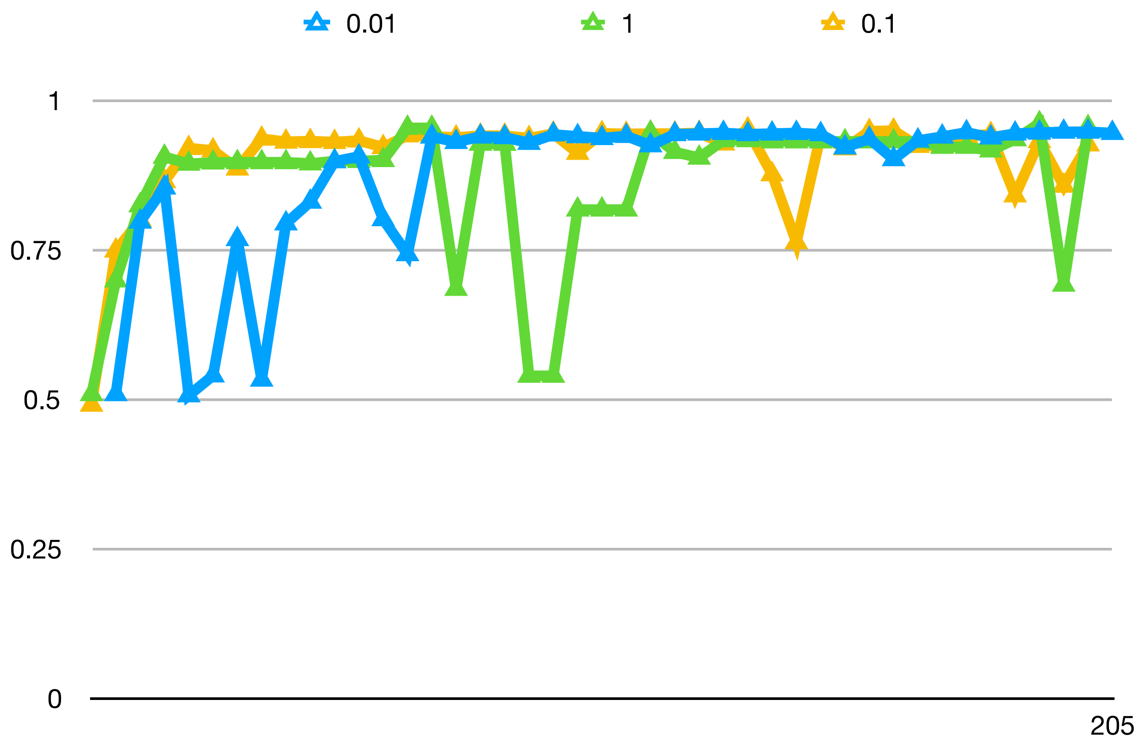
Machine Learning
CSCI 5622 Fall 2017
Name: Chi Chen

Homework 2
Due Time Sep 29, 2017
CU identity key: chch4713

1. Logistic Regression (40 pts)

I. What is the role of the learning rate (eta) on the efficiency of convergence during training?

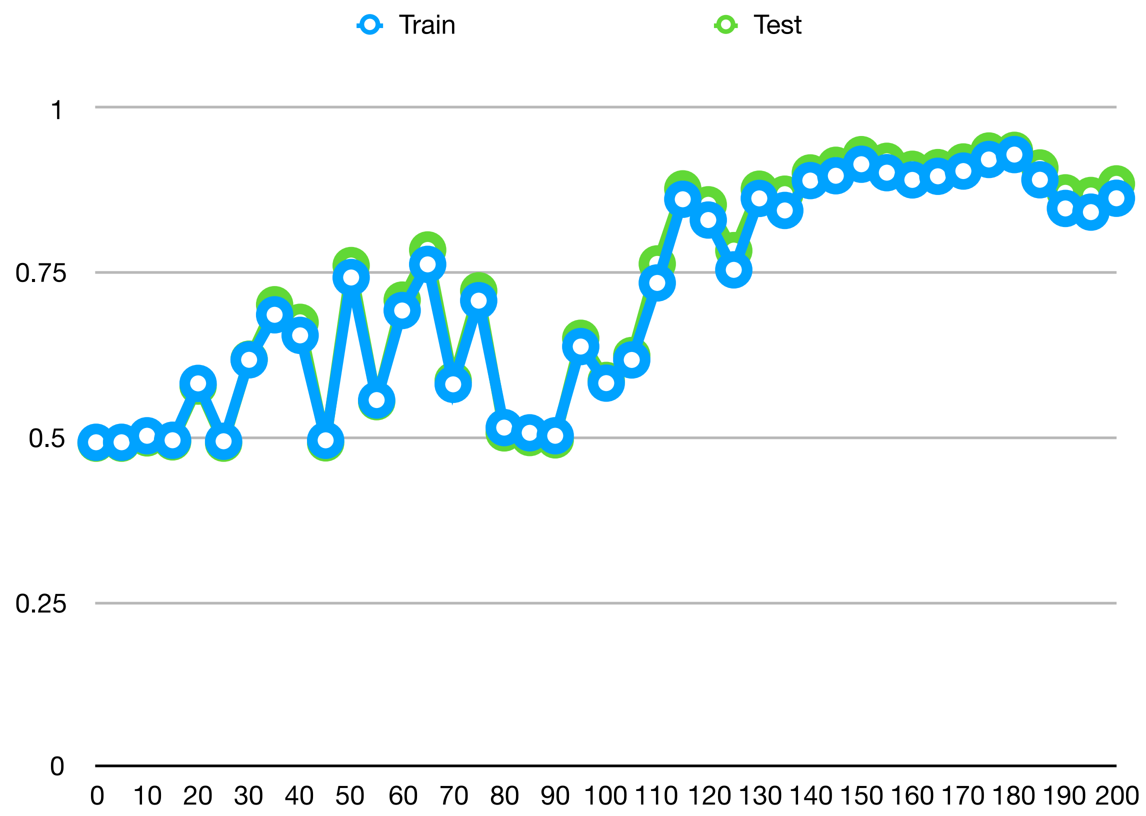
If the learning rate is too small, it will take more iterations to converge, so it will be slower. But if the learning is too larger, it may be overshooting or can't converge.



The figure shows that when the eta is 0.01, the converge speed is the slowest, but it is the most stable. And when eta is 1, the coverage speed is the fastest, but it is not that stable.

II. What is the role of the number of epochs on test accuracy?

The number of epochs will effect the accuracy of test set too. If the number of epochs were too small, the underfitting problem will appear, the train data weren't enough to train the model. But if the number of epochs were too large, the overfitting problem might also show up.

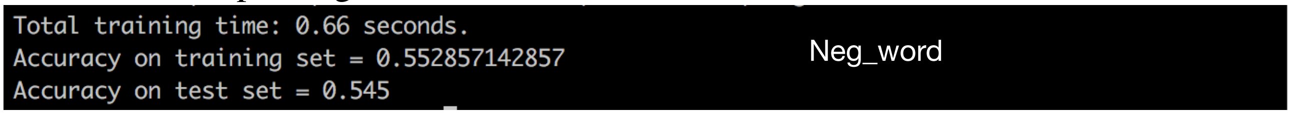


As you can see, when the number of epochs were small, the accuracy rate is not that high, when the number of epochs became more, the accuracy rate will be higher. But when the number of epochs became too high, the accuracy rate might be lower.

2. Feature Engineering (40 pts)

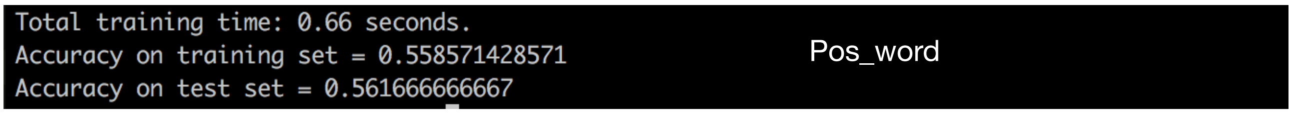
III. What custom features did you add/try (other than n-grams)? How did those additional features affect the model performance? Why do you think those additional features helped/hurt the model performance?

I added two custom features, which were Pos_word and Neg_word, they will count the numbers of pos/neg words in each movie review.



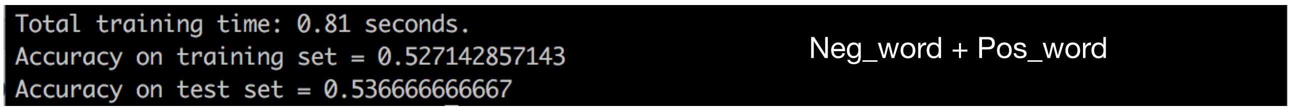
```
Total training time: 0.66 seconds.
Accuracy on training set = 0.552857142857
Accuracy on test set = 0.545
```

Neg_word



```
Total training time: 0.66 seconds.
Accuracy on training set = 0.558571428571
Accuracy on test set = 0.561666666667
```

Pos_word



```
Total training time: 0.81 seconds.
Accuracy on training set = 0.527142857143
Accuracy on test set = 0.536666666667
```

Neg_word + Pos_word

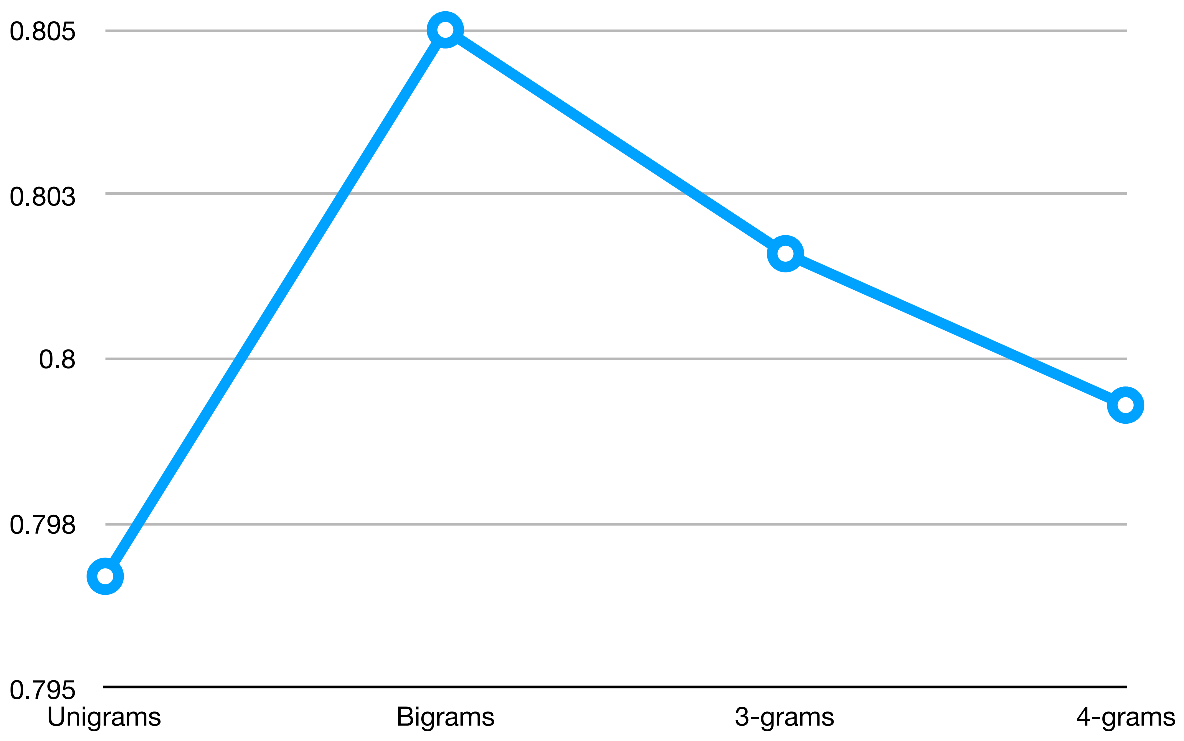
The upper figure is the performance of adding the Neg_word, and the middle one is after adding Pos_word, the latest one combined both Pos_word and Neg_word.

Since the label was related to the review, Pos_word and Neg_word could help learning the label of the review. But the weird thing was that when both Pos_word and Neg_word were used, the performance was decreasing, it might be because the model was confused by the Pos_word and Neg_word.

IV. What are unigrams, bigrams, and n-grams? When you added those features to the Feature-Union, what happened to the model performance? Why do these features help/hurt?

The unigrams means that whole review will be split into a list which contained every word in the review, the bigrams will contain every two continues words and then the n-grams will contain every n continues words. When I added this feature, the

performance increase to about 80% immediately. I used n = 1~4 to compared the performance.



This feature could analyze the content of the review and help learning about the label a lot, but the length of set will determine the performance too. As we can see, the bigrams's performance is the best, and the larger n will decrease the performance. The reason might be overfitting, when the n became too large, the overfitting will show up.

3. Gradient Descent Learning Rule for Multi-class Logistic Regression (20 pts)

We introduced binary classification using logistic regression in class. This framework can be extended to perform multi-class classification. Specifically, assuming the training set is $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, $y_i \in \{1, 2, \ldots, C\}$, independent and identically distributed with

$$p(y = c|\boldsymbol{x}) = \frac{\exp(\boldsymbol{\beta}_c^T \boldsymbol{x})}{\sum_{c'=1}^{C} \exp(\boldsymbol{\beta}_{c'}^T \boldsymbol{x})}.$$

Note: since we are doing multi-class logistic regression, we have a different weight vector $\boldsymbol{\beta}_k$ for each class c.

- Derive the negative log likelihood for multi-class logistic regression.

The negative log likelihood can be written as follow.

$$L(\beta) = -\prod_{i=1}^{N}\prod_{c=1}^{C} \mu_{ic}{}^{y_{ic}} = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic} \log \mu_{ic}$$

$$= -\sum_{i=1}^{N}[(\sum_{c=1}^{C} y_{ic}\beta_c^T x_i) - \log(\sum_{c'=1}^{C}\exp(\beta_{c'}^T x_i))]$$

- The gradient descent learning rule for optimizing weight vectors generalizes to the following form: $\beta^{t+1} = \beta^t - \eta\nabla\beta^t$ where η is the learning rate. Find the $\nabla\beta_{c,j}$ (the parameter for feature x_j in class c) for a multi-class logistic regression model.

The gradient for $\nabla\beta_{c,j}$ can be written as follow.

$$\nabla_{\beta_{yj}}(\sum_{c=1}^{C} y_{ic}\beta_c^T x_i + \log(\sum_{c'=1}^{C}\exp(\beta_{c'}^T x_i)))$$

$$= -y_{ic}x_j + \frac{x_j \exp(\beta_{y_j}^T x_j)}{\log(\sum_{c'=1}^{C}\exp(\beta_{c'}^T x_j))}$$