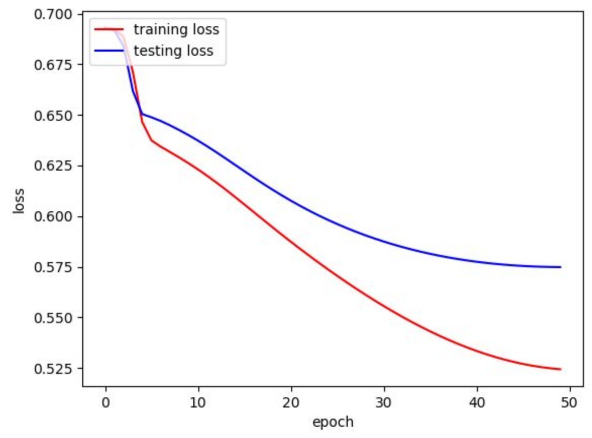
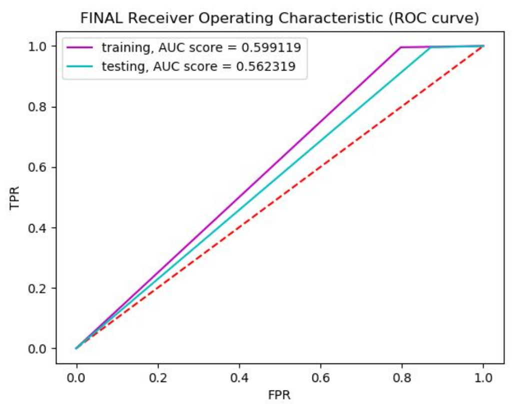
Progress Summary 8/6/18

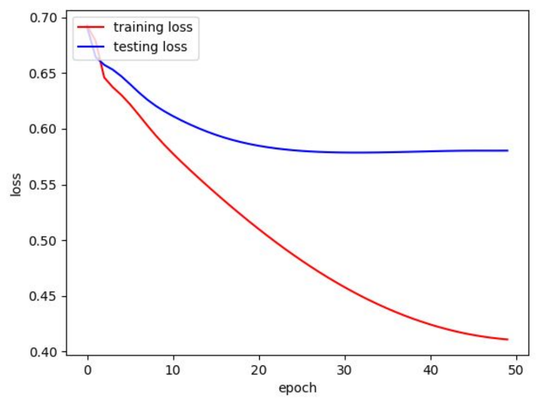
This past week I have mainly been attempting to add some sort of regularization to my model in order to reduce overfitting. I have tried several methods, and haven't really had the greatest of luck.

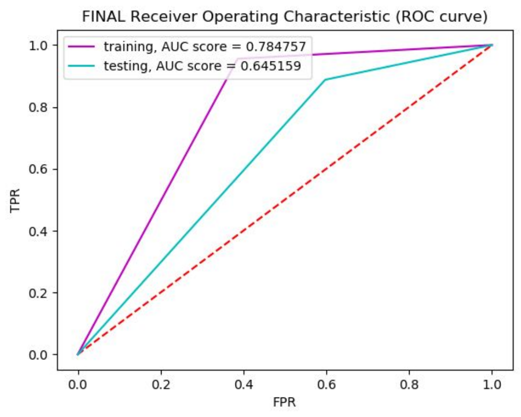
- I've tried normalizing the matrices when updating the gradient (which is what fastText seems to do). This is essentially adding a division by the total sum of the input value as the 'activation function' which is what fastText does as well as word2vec. This version is the most similar to fastText out of all of the regularization methods I've tried. Below are all of my attempts at fine tuning the parameters.



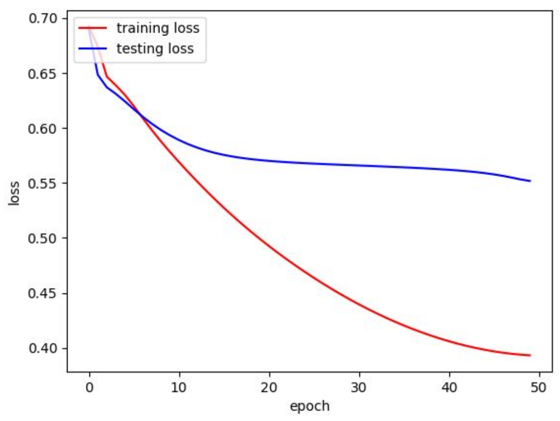


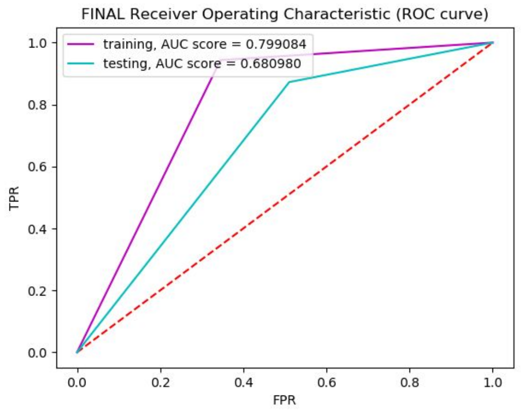
LR = 0.1, 50 epochs, 1000 instances



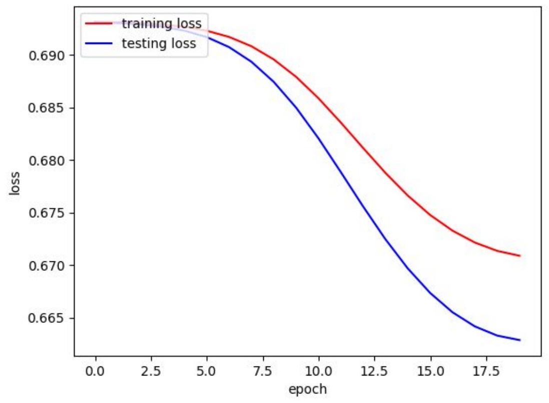


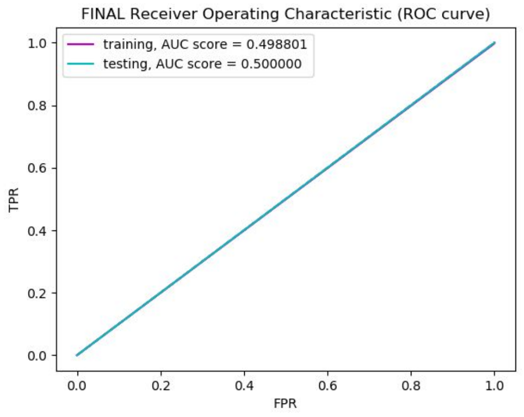
LR = 0.1, epochs 50, 2000 instances



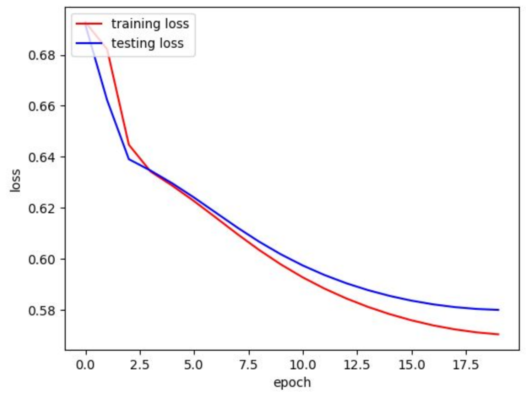


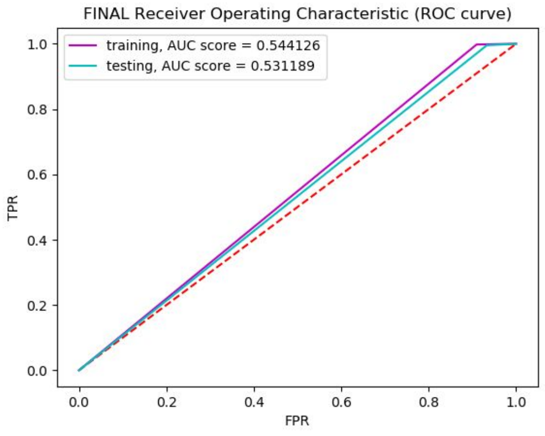
LR = 0.09, epochs 50, 3000 instances





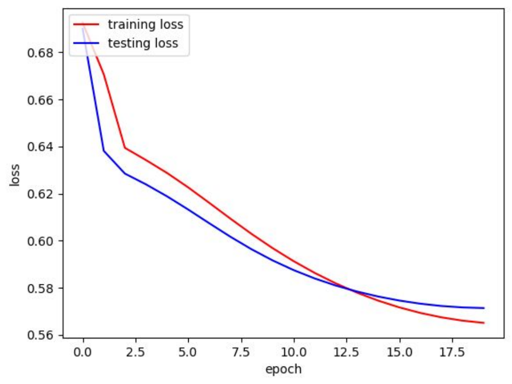
LR = 0.05, epochs 20, 1000 instances

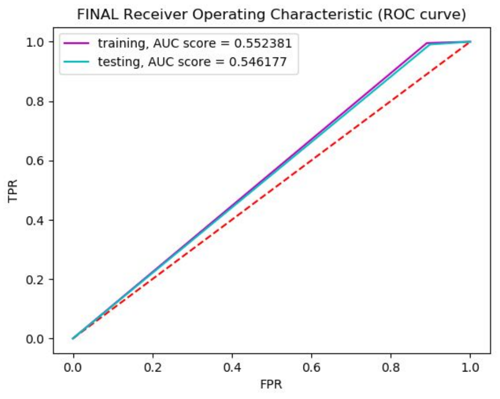




LR = 0.05, epochs 20, 4000 instances

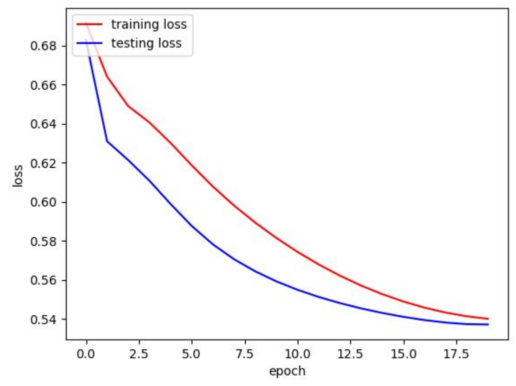
```
EPOCH:  19
Train:    [0.57039443]
Test:     [0.57997697]

confusion matrix: TRAIN
[  82 824  ]
[  4 1770  ]
AUC score:  0.5441264674187616

confusion matrix: TEST
[  30 418  ]
[  4 868  ]
AUC score:  0.5311885648754915


TRAIN:
        Classification Err:  0.30895522388059704
        Precision:           0.6823438704703161
        Recall:              0.9977452085682075
        F1:                  0.8104395604395604
TEST:
        Classification Err:  0.3196969696969697
        Precision:           0.6749611197511665
        Recall:              0.9954128440366973
        F1:                  0.8044485634847081
```

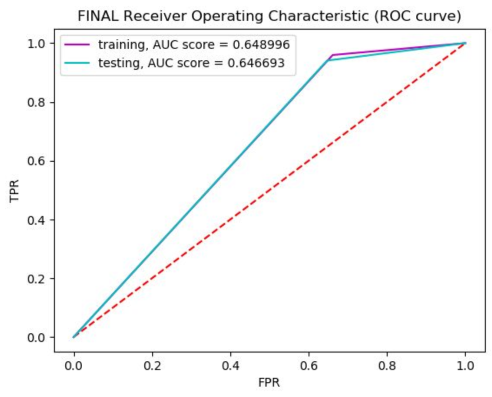




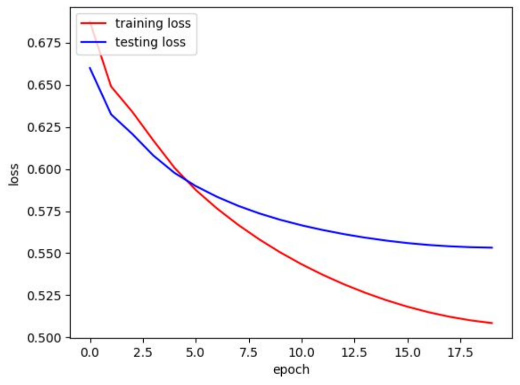
LR = 0.06, epochs 20, 4000 instances

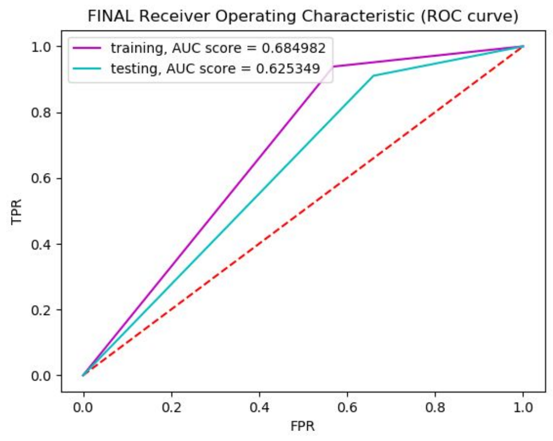
```
EPOCH:  19
Train:    [0.56512278]
Test:     [0.57140808]

confusion matrix:
[  99 807  ]
[  8 1766  ]
AUC score:  0.5523809701576115

confusion matrix:
[  44 390  ]
[  8 878  ]
AUC score:  0.5461765715534012


TRAIN:
        Classification Err:  0.3041044776119403
        Precision:           0.6863583365720949
        Recall:              0.9954904171364148
        F1:                  0.8125143777317689
TEST:
        Classification Err:  0.3015151515151515
        Precision:           0.692429022082019
        Recall:              0.9909706546275395
        F1:                  0.8152274837511606
```





LR = 0.08, epochs 20, 4000 instances

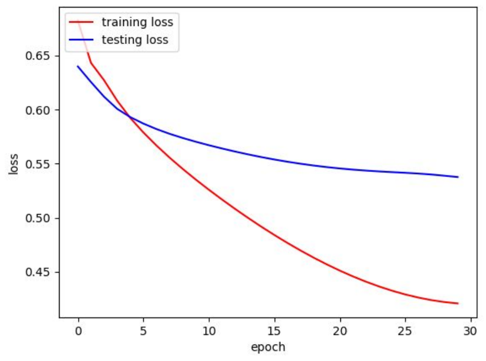


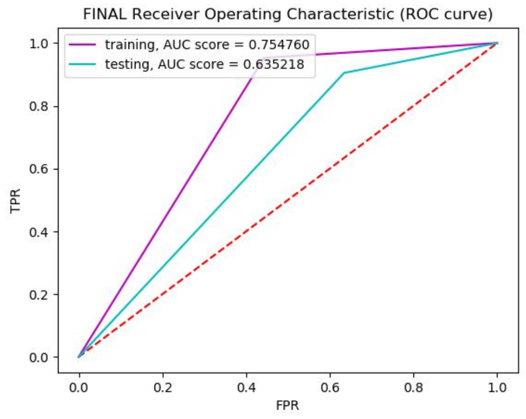


Lr = 0.1, 20 epochs, 4000 instances

```
EPOCH:  19
Train:    [0.50851886]
Test:     [0.55328072]

confusion matrix:
[  412 543  ]
[  106 1619  ]
AUC score:  0.6849821686015631

confusion matrix:
[  152 295  ]
[  78 795  ]
AUC score:  0.6253488318457529
```

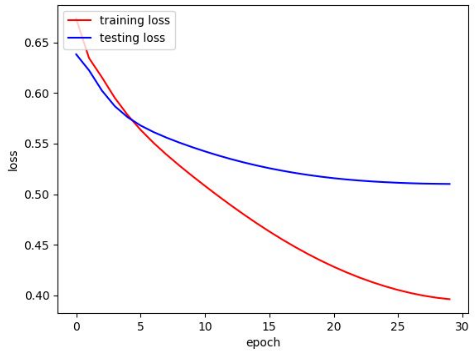
```
TRAIN:
        Classification Err:  0.2421641791044776
        Precision:           0.7488436632747456
        Recall:              0.9385507246376812
        F1:                  0.8330331875482376
TEST:
        Classification Err:  0.2825757575757576
        Precision:           0.7293577981651376
        Recall:              0.9106529209621993
        F1:                  0.8099847172694854
```

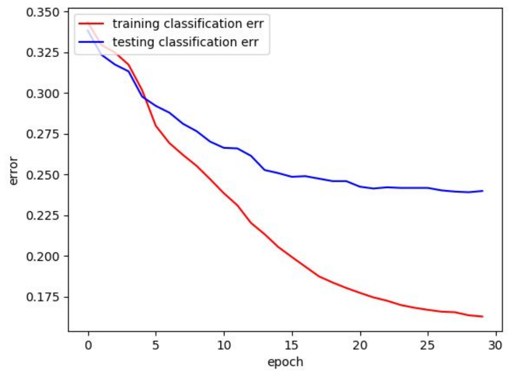


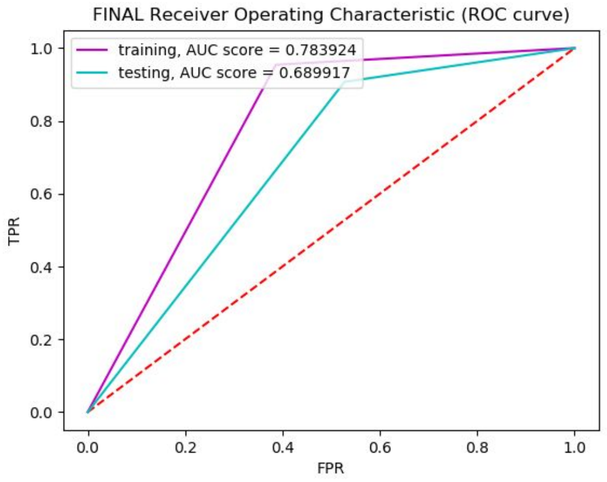


LR = 0.1, epochs 30, 5000 instances

```
EPOCH:  29
Train:    [0.4208916]
Test:     [0.53774018]

confusion matrix:
[  652 519  ]
[  103 2076  ]
AUC score:  0.7547598397716891

confusion matrix:
[  200 347  ]
[  105 998  ]
AUC score:  0.6352178950212235


TRAIN:
        Classification Err:  0.18567164179104478
        Precision:          0.8
        Recall:             0.9527306103717301
        F1:                 0.8697109342270632
TEST:
        Classification Err:  0.2739393939393939
        Precision:          0.7420074349442379
        Recall:             0.9048050770625566
        F1:                 0.8153594771241831
```





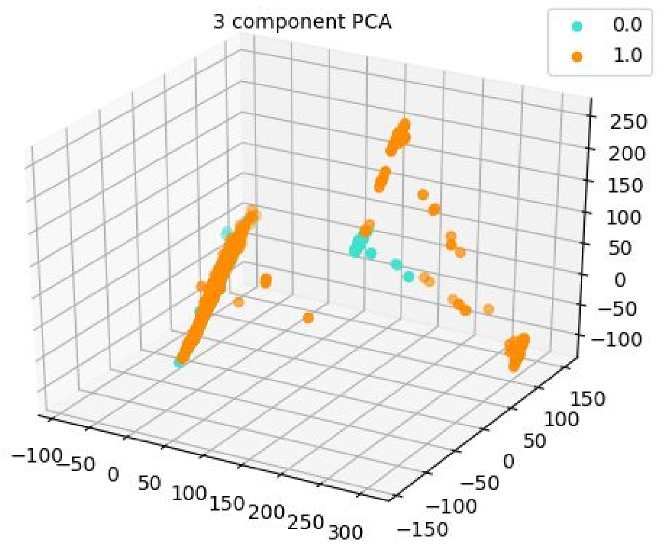


LR = 0.08, epochs = 30, 8000 instances

```
EPOCH:  29
Train:    [0.39634472]
Test:     [0.5101549]

confusion matrix:
[  1129 711  ]
[  161 3359  ]
AUC score:  0.7839241600790514

confusion matrix:
[  422 471  ]
[  162 1585  ]
AUC score:  0.6899169973674275


TRAIN:
        Classification Err:  0.1626865671641791
        Precision:          0.8253071253071254
        Recall:             0.9542613636363636
        F1:                 0.8851119894598155
TEST:
        Classification Err:  0.23977272727272728
        Precision:          0.7709143968871596
        Recall:             0.9072696050372067
        F1:                 0.8335524585853273
```

- I also attempted to implement dropout with this method, which didn't reduce overfitting by much.
- Another method to reduce overfitting was adding a 'quadratic regularization' term, which I found in this paper. https://arxiv.org/pdf/1804.00306.pdf. But I didn't get very good results with that either.
- I also added implemented dropout with the above method, which also didn't work well.

I also implemented a script that uses PCA to plot the data in 3D (or 2D) space, in order to visualize the distribution of the data. I was thinking of plotting the distribution of the unlabeled dataset as well in order to see the two dataset's distributions, in order to verify that they are in fact, from different distributions. *But is there an easier way to verify this? Like some sort of mathematical equation that compares data distributions?*



8000 instances.

From reading the paper “Learning from Labeled and Unlabeled Data,” I believe that the dataset I am using (the automatically labeled dataset) is of type MNAR (missing not at random) because there is a “sample-selection bias in constructing the labeled set that has to be corrected (332).” I believe that there is a sample-selection bias when constructing this dataset because on page 341, it says “If there is a censoring mechanism that rules the assignment of a label to instances in the data, then a model learned on only the labeled data will have a sample selection bias.” And I believe that labeling data based on the specific heuristics that were used (ie. “im a man,” = male label, “im a woman” = female label, etc.) created a sampling selection bias since a specific subset of the population will be chosen to generate this dataset.

This paper suggests two ways to deal with sample selection bias (see page 341)

- Bivariate Probit
- Sample-Select

I am just wondering where I should go from here? Do you agree that the data is of MNAR type? Should I try to implement both versions? I am also struggling to find a way to measure the bias of the model.

I am also wondering how similar my model needs to be to the fastText model? Does it need to be exactly the same as fastText, or can there be a few things that are slightly different? For example, fastText takes as input character ngrams, but if I were to include these, the dimensions would become extremely large, and training on my computer will take an extremely long time, and I think the accuracy might suffer as well with my implementation. Do you think it’s fine to leave out the character ngrams? Or should I include them since they are part of the fastText implementation?

I am also getting worried that I'm falling too far behind. Is the progress I'm making okay so far?

Research Strategy for this week:

- Implement either Bivariate probit or sample-select (or both?)
- I will need to create a web scraper to collect the new unlabeled dataset from Twitter.