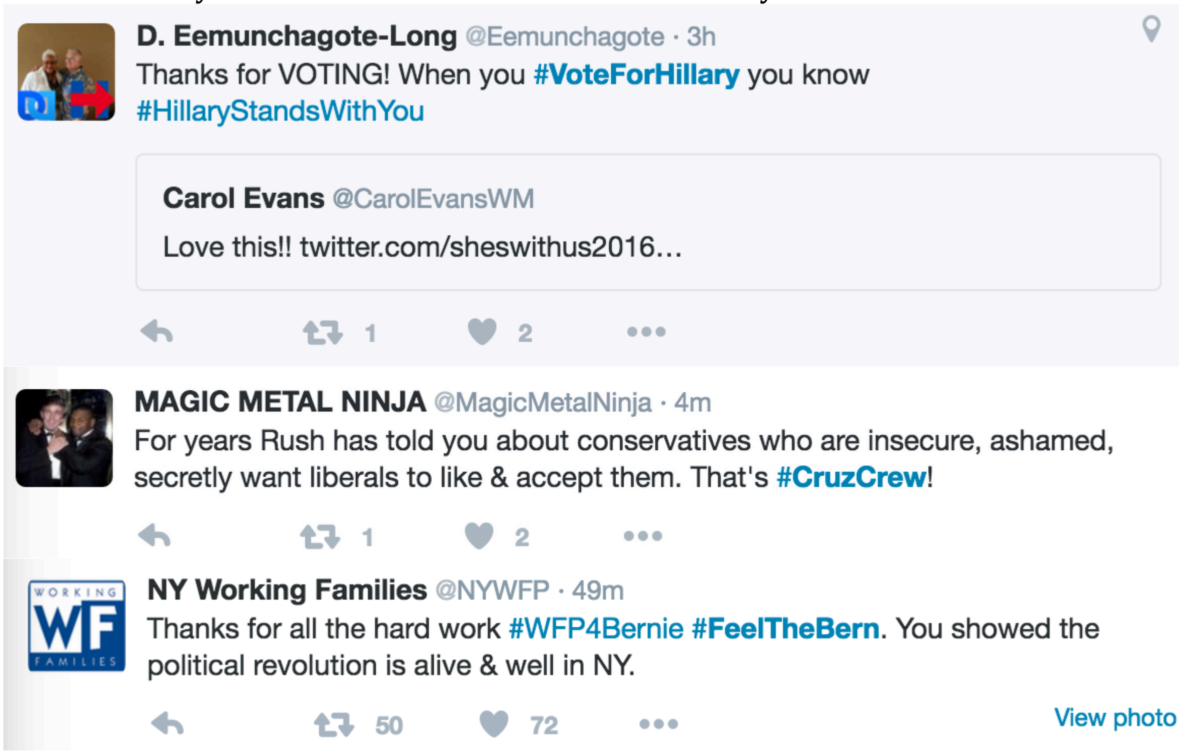
Predicting twitter for 2016 Presidency Election

Introduction:
In this project, I plan to use twitter API to grab some tweets with political tendency as my training data (for example, grab tweets with topics like "#voteforhillary" and so on) and use the same type of unseen tweets but different with training set as my testing data (and delete the hashtag). After that, I will use bag of words algorithm and different classifier to compare the running time and accuracy, decide which model is the best and use this model to predict people's tendency of voting (get one's all twitter and run though my model).

Dataset source:
Using Twitter API to grab tweets online.

```
tweepy.Cursor(api.search, q='voteforhillary', since='2015-04-01', until='2016-04-09').
```

Training:
Grab with hash tags, and use hash tags to do the classification. For example, hash tag #voteforhillary means this tweet want to vote for Hillary Clinton.



D. Eemunchagote-Long @Eemunchagote · 3h
Thanks for VOTING! When you #VoteForHillary you know #HillaryStandsWithYou

Carol Evans @CarolEvansWM
Love this!! twitter.com/sheswithus2016…

1 2

MAGIC METAL NINJA @MagicMetalNinja · 4m
For years Rush has told you about conservatives who are insecure, ashamed, secretly want liberals to like & accept them. That's #CruzCrew!

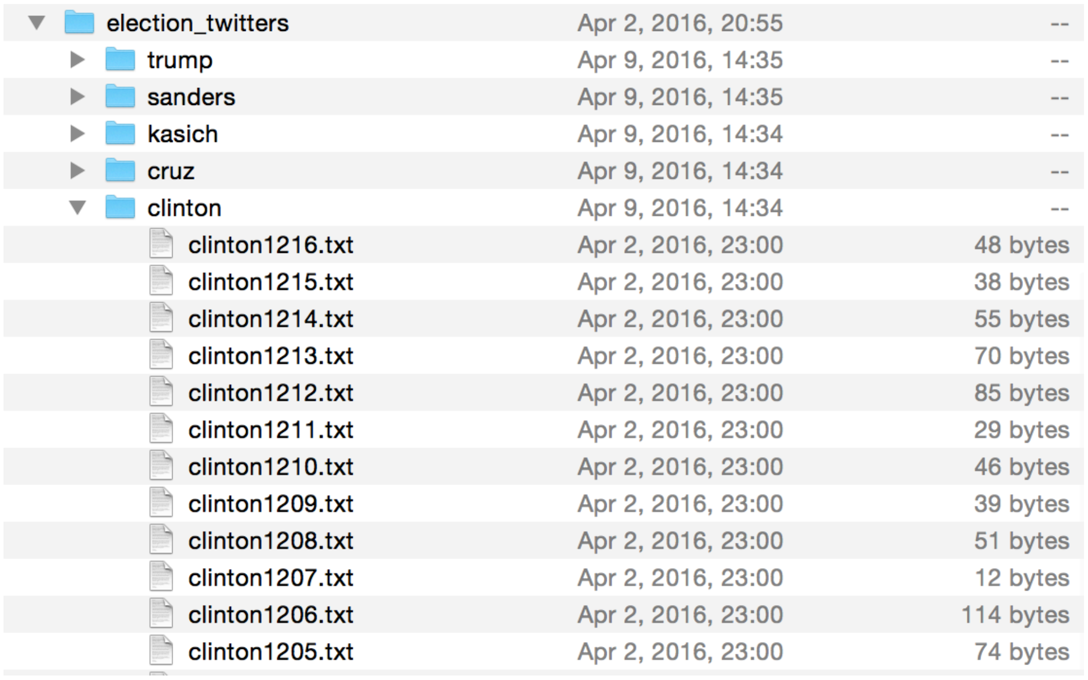
1 2

NY Working Families @NYWFP · 49m
Thanks for all the hard work #WFP4Bernie #FeelTheBern. You showed the political revolution is alive & well in NY.

50 72 View photo

Testing:
Grab different tweets and delete the hash tags. Since not every tweet would have obvious tendency for voting, I delete the hash tags and keep the content for predicting(I know the classes of each tweets but my program does not, so it is a fine way to test my algorithm).

The files as follow:



election_twitters	Apr 2, 2016, 20:55	--
trump	Apr 9, 2016, 14:35	--
sanders	Apr 9, 2016, 14:35	--
kasich	Apr 9, 2016, 14:34	--
cruz	Apr 9, 2016, 14:34	--
clinton	Apr 9, 2016, 14:34	--
clinton1216.txt	Apr 2, 2016, 23:00	48 bytes
clinton1215.txt	Apr 2, 2016, 23:00	38 bytes
clinton1214.txt	Apr 2, 2016, 23:00	55 bytes
clinton1213.txt	Apr 2, 2016, 23:00	70 bytes
clinton1212.txt	Apr 2, 2016, 23:00	85 bytes
clinton1211.txt	Apr 2, 2016, 23:00	29 bytes
clinton1210.txt	Apr 2, 2016, 23:00	46 bytes
clinton1209.txt	Apr 2, 2016, 23:00	39 bytes
clinton1208.txt	Apr 2, 2016, 23:00	51 bytes
clinton1207.txt	Apr 2, 2016, 23:00	12 bytes
clinton1206.txt	Apr 2, 2016, 23:00	114 bytes
clinton1205.txt	Apr 2, 2016, 23:00	74 bytes

There are more than 50000 tweets in total. 35000+ training data and 20000+ testing data.

Bag of words(pre-processing):

The bag-of-words model is a simplifying representation used in natural language processing and information retrieval (IR). In this model, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity.

Step 1:
Go through all the data, and create a dictionary which contains all the words appeals in the twitters and not in the stop words list, and save the dictionary as W.
Step 2:
Go through the data again, for each tweet, count the words frequency in the order of dictionary W. Therefore, after go through all tweets, we would get a M*N matrix which M is corresponding each tweets and the class label, while N records the word frequency in each tweets.
Step 3:
Use this matrix as our training data.

Models:
Naive Bayes and Neural Network.

Naïve Bayes:

Naïve Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable.

In this case, I used naiveBayesClassifier package to compute Naïve Bayes algorithm, and I used every words in the dictionary as my attributes to train the model, and use the model to predict the class which has the highest conditional probability given the content of the testing tweets.

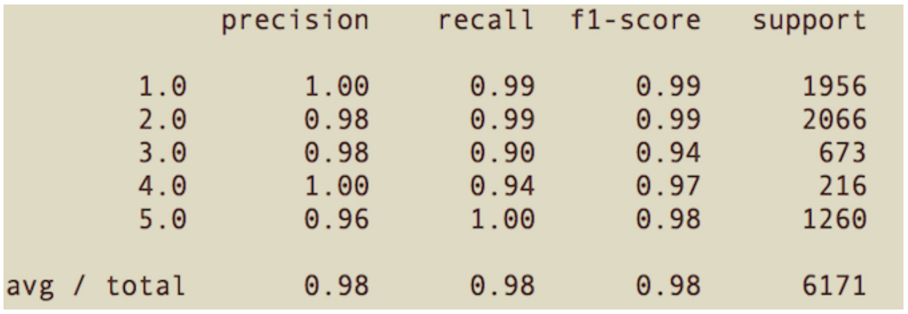
Neural Network:

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurones. This is true of ANNs as well.

In this case, I used scikit-learn package(sklearn.neural_network) to compute neural network algorithm, and I used every words in the dictionary as my attributes to train the model. I set the number of hidden units is 100(1 layer) and iteration time is 20 (and I tried much larger number of these numbers and it has no influence to the result, which means this package has some inner mechanism to avoid overfitting.)

The reason to use these two model is Naïve Bayes has fastest run time and good accuracy and Neural Network has better accuracy, and both of them are good at bag of words and easy to program.
Naïve Bayes runs few seconds and Neural Network runs about 40 min.

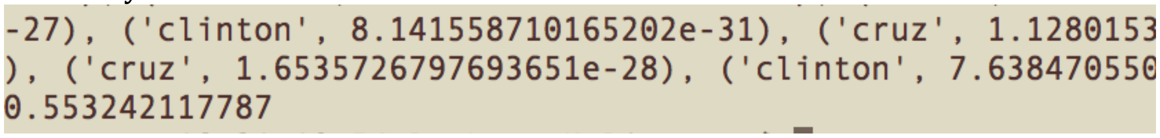
This gives us some feelings that if we want to predict in a fast way, we can use Naïve Bayes model, and if we want to keep our models update once new data comes or keep the model to do the deep learning, we should use the neural network model.

Result:
Accuracy:
With hash tags:
Naïve Bayes: 92%
Neural Network: 98%


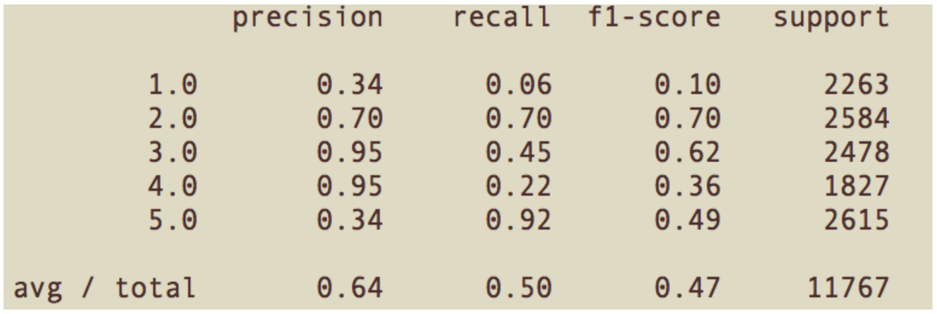
```
             precision    recall  f1-score   support

        1.0       1.00      0.99      0.99      1956
        2.0       0.98      0.99      0.99      2066
        3.0       0.98      0.90      0.94       673
        4.0       1.00      0.94      0.97       216
        5.0       0.96      1.00      0.98      1260

avg / total       0.98      0.98      0.98      6171
```

Without hash tags:
Naïve Bayes: 55%


```
-27), ('clinton', 8.141558710165202e-31), ('cruz', 1.1280153
), ('cruz', 1.6535726797693651e-28), ('clinton', 7.638470550
0.553242117787
```

Neural Network: 64%


```
             precision    recall  f1-score   support

        1.0       0.34      0.06      0.10      2263
        2.0       0.70      0.70      0.70      2584
        3.0       0.95      0.45      0.62      2478
        4.0       0.95      0.22      0.36      1827
        5.0       0.34      0.92      0.49      2615

avg / total       0.64      0.50      0.47     11767
```

Outputs are in the folder “Outputs”. Naïve Bayes has better run time and Neural network has better accuracy, and both of them has better result than the human judgment (My friends predict the tweets and get the accuracy of slightly over 50%. He predicted 100 tweets and got 51 right).

Summary:

Both models have very good result but I think it could be better if I use bagging or boosting to do combine a new model, or using LDA to do the prediction. That requires further study.