Michael Baldwin

Josh Engelsma

Adam Terwilliger

February 16, 2016

CIS 678 – Machine Learning

Project 2

**Abstract**

We look to further expand our abilities in CIS 678 – Machine Learning, through learning and implementing the Naïve Bayes algorithm for document classification. Using Python, we were able to develop a supervised learning model that uses probabilities from Bayes Theorem. As such, we abstracted our code in a way that allowed for modularity to train and test two different types of datasets: forums and twitter. Using three different validation approaches, we found classification rates over 80% for forum data (guessing would be 1/20 = 5%) and 75% for twitter data (guessing would be ½ = 50%). Our final analyses looked to demonstrate the predictive abilities of our classifier by predicting the sentiment of nearly 30 twitter users’ last 3000 tweets, and as such, we obtained a “positivity” rating for each user.

**Implementation details**

Our program is written in Python 2.7 and bash scripting in Unix. These programs were executed locally on each member’s respective Macbook Pro (2012), testing on eos23 and okami.

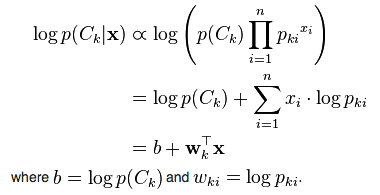
**Summary of Problem**

Naïve Bayes is a supervised learning approach that utilizes Bayes Theorem as seen in Equation 1. I should be noted that Naïve Bayes makes the simplifying assumption that features are conditionally independent. In Laymen’s terms, Naïve Bayes looks to the prior (proportion of class size to total corpus size) and the likelihood (proportion of word occurrence for particular document type). Additionally, we implement, seen in Equation 2, a log probability transformation to avoid “underflow”, rather, avoid multiplying decreasingly small probabilities together which trend to zero.





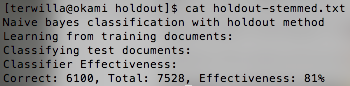
**Equation 1. Formula of Bayes Theorem**



**Equation 2. Naïve Bayes log transformation**

**Results**

Our model correctly classifies over 81% of forum documents stemmed with Porter’s Stemmer using a 60/40 training/test holdout validation method, as we note in Figure 1.



**Figure 1. Sample validation output using original holdout split**

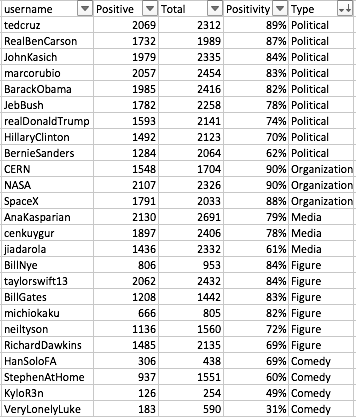
Using the k-fold validation approach, we found 12-fold (92/8) with 86% classification rate offered the most promising results. We find Figure 3, maximizes over the k iterations with the maximum over 2 through 15-fold validation landing around 80%.

The randomized validation approach proved to be the most effective, as we observe in Figure 3. We found nearly 89% classification rate using a 78/22 training/test split for the forum-stemmed data. We explored a sentiment analysis case-study training our Naïve Bayes classifier using over 1.5 million tweets pre-labeled with positive or negative sentiment. Using the randomized validation approach for this twitter data, we observed a classification rate of nearly 76% using an 87/13 training/test split.

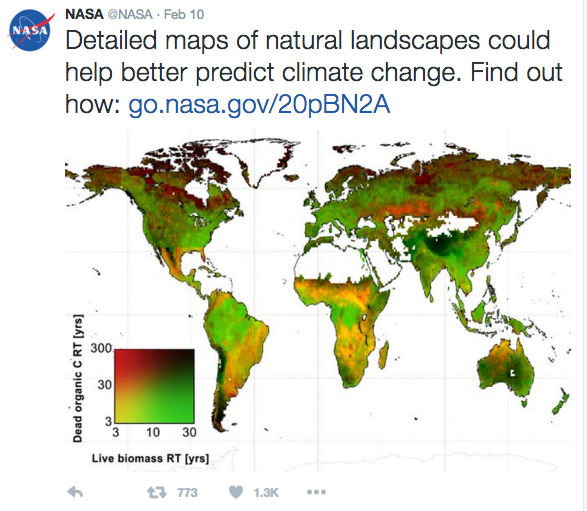
**Discussion**

One interesting feature we find in Figure 3 is missing value imputation. In our original dataset, 7 of the 744 total data points were missing. As such, we imputed these values with the quadratic predicted values for number of downloads. We choose this model for the imputation over the cubic with the principle of balancing model simplicity with the amount of variation explained.

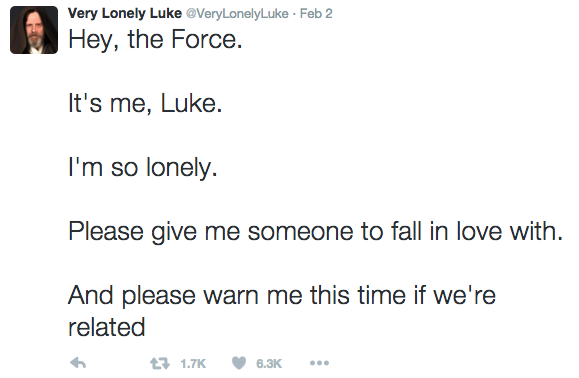
We began to extract features of the dataset in Time of Day, Day of Week, and Day of Month as seen in Figures 4, 5, and 6; respectively.



**Table 1. R-Squared values for First, Second, and Third order models**



**Figure 5. Example of Positive Tweet.**

****

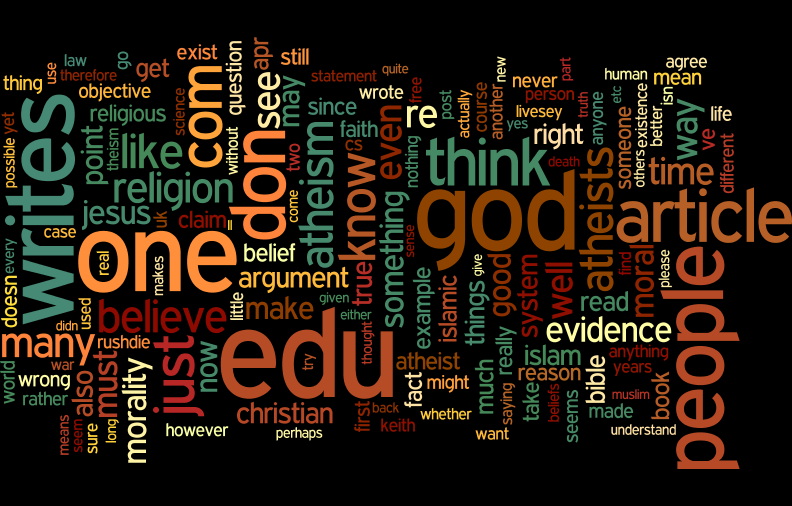
**Figure 6. Example of Negative Tweet.**

Figure 4 provides little to no additional information, as we can infer that Time of Day would not be a valuable feature in a multiple regression model due to equal variance throughout the day with only a slight peak around 4/5 pm. Additionally, Figure 5 shows a great peak on days 1, 2, and 3 (we did not have day markers i.e. Sunday, Monday, etc.). However, this is a result of having 1 additional day contributing to the average downloads for the day, with the first three days showing the effect of the rise in downloads at the end of the month, as seen in Figure 6.

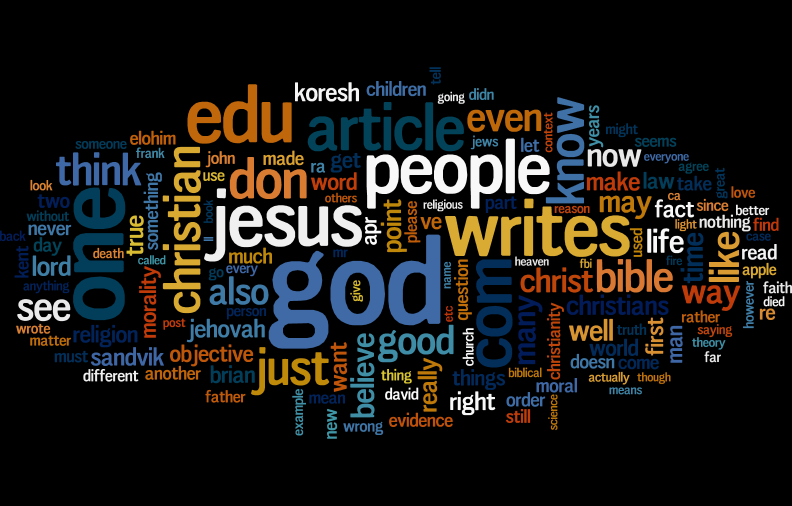
Our final note is with regards to avoiding overfitting the model as we may be encouraged by a higher order model explaining more of the variation in downloads; however, in future work, we should apply appropriate machine learning techniques of training and test sets to avoid this issue.

**Future Work**

We have four main directions we would pursue if time allowed: topic clustering, precision/recall, n-grams, and maximum entropy. We can gain some preliminary intuition from the word clouds in Figures 7 and 8 that topics like “atheism” and “religion” may be quite similar as we note words like “god”, “people”, “belief” and “faith” appear frequently in both classes of documents.

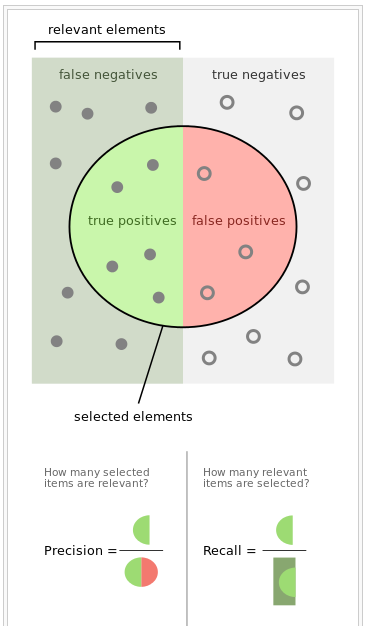


**Figure 7. Atheism Word Cloud**



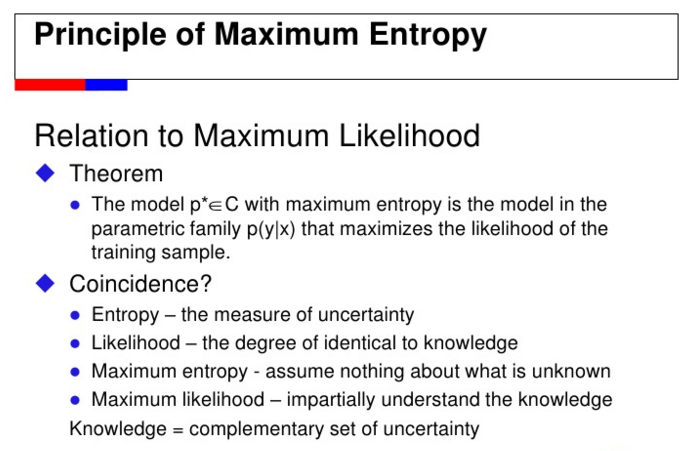
**Figure 8. Religion Word Cloud**

As a complementary analysis to traditional training vs. testing validation, precision/recall offers additional insights into the types of error that the classifier is making, as seen in Figure 9.

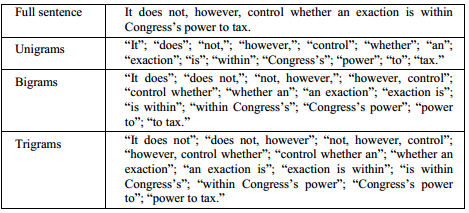


**Figure 9. Precision vs. Recall**

As mentioned in the summary of the problem, Naïve Bayes makes the underlying assumption that features/observations are independent. Maximum Entropy classification and n-grams look to an alternative, as we may find instances where independence may not be inferred (i.e. “President”, “Obama” / “President”, “Bush” vs. “President Obama” / “President Bush”). We can understand more about maximum entropy and n-grams in Figures 10 and 11.



**Figure 10. Further exploration of Maximum Entropy Classification**



**Figure 11. Example of n-grams**

**Credits**

- [Simple Explanation of Naive Bayes](http://stackoverflow.com/questions/10059594/a-simple-explanation-of-naive-bayes-classification)

- [Where to start with text mining](http://tedunderwood.com/2012/08/14/where-to-start-with-text-mining/)

- [Intro to Topic Modeling](http://journalofdigitalhumanities.org/2-1/topic-modeling-a-basic-introduction-by-megan-r-brett/)

- [Naive Bayes Time Complexity](http://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html)

- [K-fold Cross Validation](https://www.cs.cmu.edu/~schneide/tut5/node42.html)

- [Python - Time Complexity of Operations](https://www.ics.uci.edu/~pattis/ICS-33/lectures/complexitypython.txt)

- [Python Progress Bar](https://github.com/WoLpH/python-progressbar)

- [Python K-fold Cross Validation](<http://stackoverflow.com/questions/16379313/how-to-use-the-a-10-fold-cross-validation-with-naive-bayes-classifier-and-nltk)>

*Important pre-processing code for twitter data was imported with all credit to yogeshg.*

- [Twitter-sentiment] (<https://github.com/yogeshg/Twitter-Sentiment)>

*Using the Twitter API, all credit to tweet scraping goes to yanofsky and tweepy.*

- [Twitter for Python] (<https://gist.github.com/yanofsky/5436496>, <http://www.tweepy.org/)>

*All stemming and removing stop words gives credit to mchaput’s Porter’s stemmer library.*

- [Stemming] (<https://bitbucket.org/mchaput/stemming)>