# CIS 365 Project 3: AI Applications

## Names: Gabriel Stepanovich and Adam Terwiliger

### Due Date

- at the start of class on Tuesday, April 19.

### Before Starting the Project

- Read the entire project description before starting. A team must consist of exactly two students and use a pair programming approach.

### Learning Objectives

After completing this project you should be able to implement AI algorithms in the areas of: NLP, computer vision, modeling and simulation, robotics, or a similar AI concept.

### Rubric

10 pts presentation ________________
10 pts elegant source code ________________
25 pts concise design ________________
25 pts notable implementation ________________
30 pts results during testing ________________

### Step 1: Select and develop ONE of the following four options:

Option 1) Use OpenCV or a similar open source software to implement a CV application

- Develop an application that can solve an identification or tracking problem. As an example you might develop software to 1) recognize insect, plant, or vehicle make or 2) track a ball, human, or animal.

Option 2) Develop an NLP agent to analyze documents at a real time pace

- Use a pre-defined corpus or domain, e.g., Twitter, FB, Wikipedia, poetry, or Shakespeare.
- You might develop an agent to determine the polarity, sentiment, sarcasm, ontology mappings (e.g., Cyc, SUMO, WordNet, SENSUS), clusters of documents, and documents on a given topic.
- You might alternatively develop an agent to compete in the Loebner Prize.

Option 3) Develop an agent for robotics

- Design an agent to utilize a LEGO Mindstorm education kit (product 9797) to complete a function, e.g., event detection, object identification, survivor location, or navigation.

Option 4) Develop an agent for gaming

- Design an agent to play against a human user, test the human's intelligence, or an environment that learns from the user's behavior.

**Step 2: Bundle your program**

- Discuss your exact project with the professor and have it approved before March 24th.
- Bundle your program and turn in all files required to run your program. Also include a readme file with explicit, detailed steps of everything needed to install, setup, and test your program.
- Ensure you include suitable documentation for your 1) overall project and 2) source code.
- Present your work the last week (Tuesday or Thursday) of class.

**Grading Criteria**

- There is a 50% penalty on programming projects if your solution does not execute or generates errors.
- There is a 50% penalty for not turning in a hardcopy (code and 1st page of this document) and softcopy (zip) to blackboard.
- Any options/approaches/requirements not specified in this document are left for your own decision making, in keeping with the spirit of the assignment.

**Late Policy**

Projects are due at the START of the class period and not accepted later. Not turning in the hard copy or soft copy by the due date is considered a late/missing project unless PRIOR arrangements are made.

## Abstract

We look to further expand our abilities in CIS 365 – Artificial Intelligence, 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 a handful of different datasets. Using a three different validation approaches: k-fold, random training/test split, and holdout, we consistently found strong results (>80% correct classification). 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 obtain 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 "arithmetic underflow", rather, multiplying decreasingly small probabilities together which trend to zero.

$$p(C_k|\mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x}|C_k)}{p(\mathbf{x})}$$

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

**Equation 1. Formula of Bayes Theorem**

$$\begin{aligned}\log p(C_k|\mathbf{x}) &\propto \log\left(p(C_k)\prod_{i=1}^{n} {p_{ki}}^{x_i}\right) \\ &= \log p(C_k) + \sum_{i=1}^{n} x_i \cdot \log p_{ki}\end{aligned}$$

**Equation 2. Naïve Bayes log transformation**

## Dataset Description

Forum – ~12,500 responses pre-trained with 20 topics (politics, religion, sports, etc.)

Twitter sentiment – ~100,000 tweets pre-trained assigned a sentiment (positive/negative)

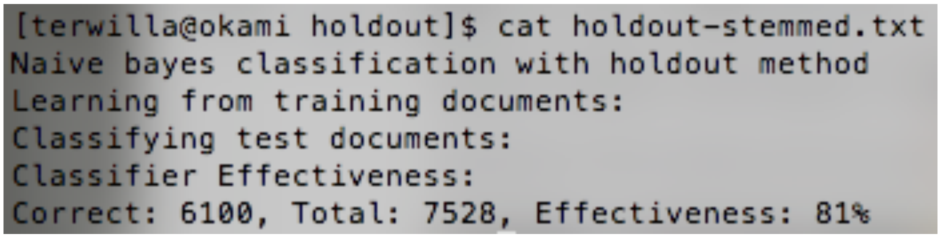
Debates – Every debate response from 2016 presidential debates (~2000) assigned an affiliation (Republican/Democrat)

Twitter categories – ~60,000 tweets assigned a category (sports, media, singers, science)

Twitter politics – ~20,000 tweets assigned Democrat or Republican

## 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.



```
[terwilla@okami holdout]$ cat holdout-stemmed.txt
Naive bayes classification with holdout method
Learning from training documents:
Classifying test documents:
Classifier Effectiveness:
Correct: 6100, Total: 7528, Effectiveness: 81%
```

**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%.

Figure 2. K-fold Validation of Naive Bayes Document Classifier

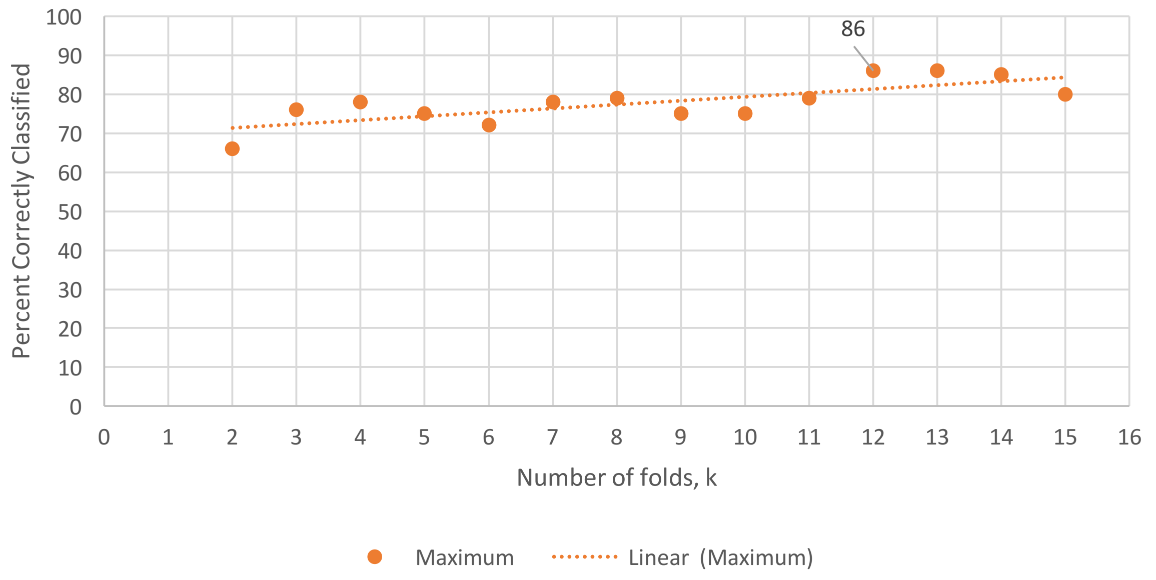
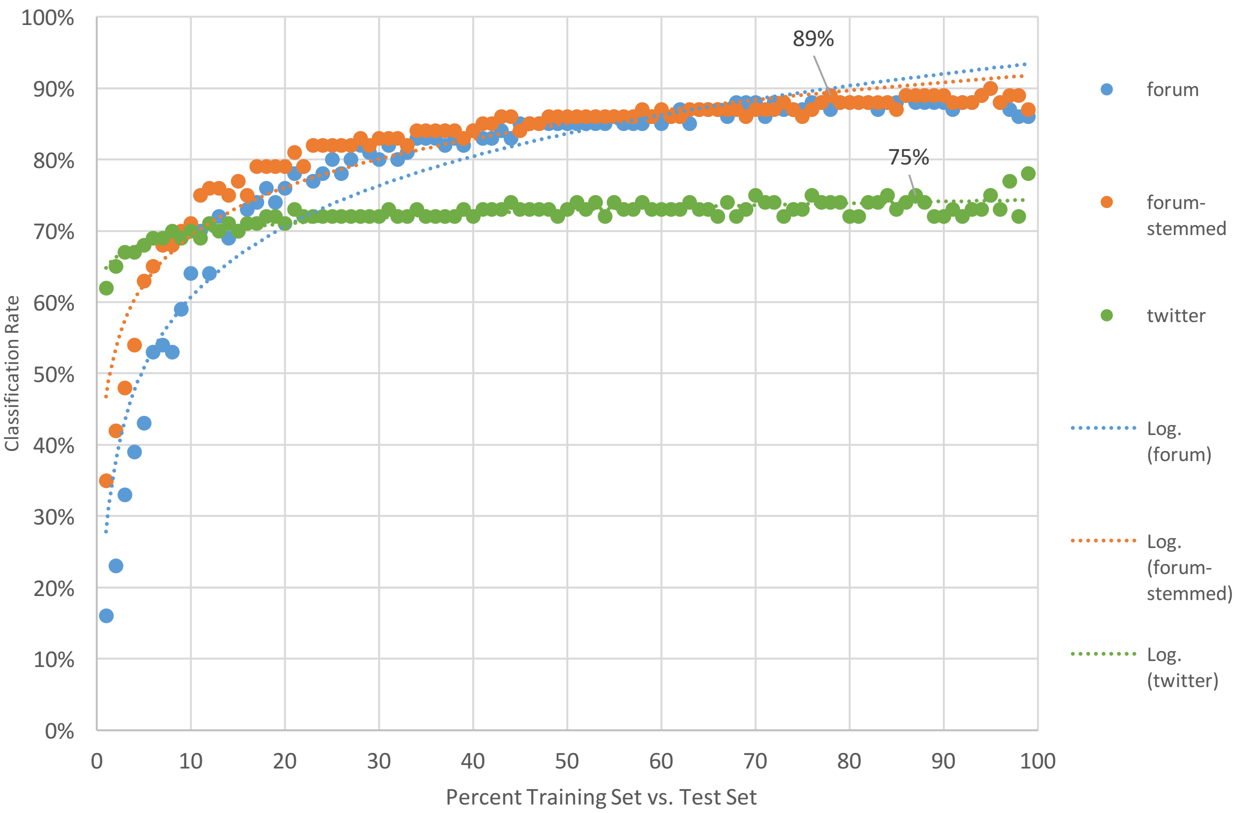
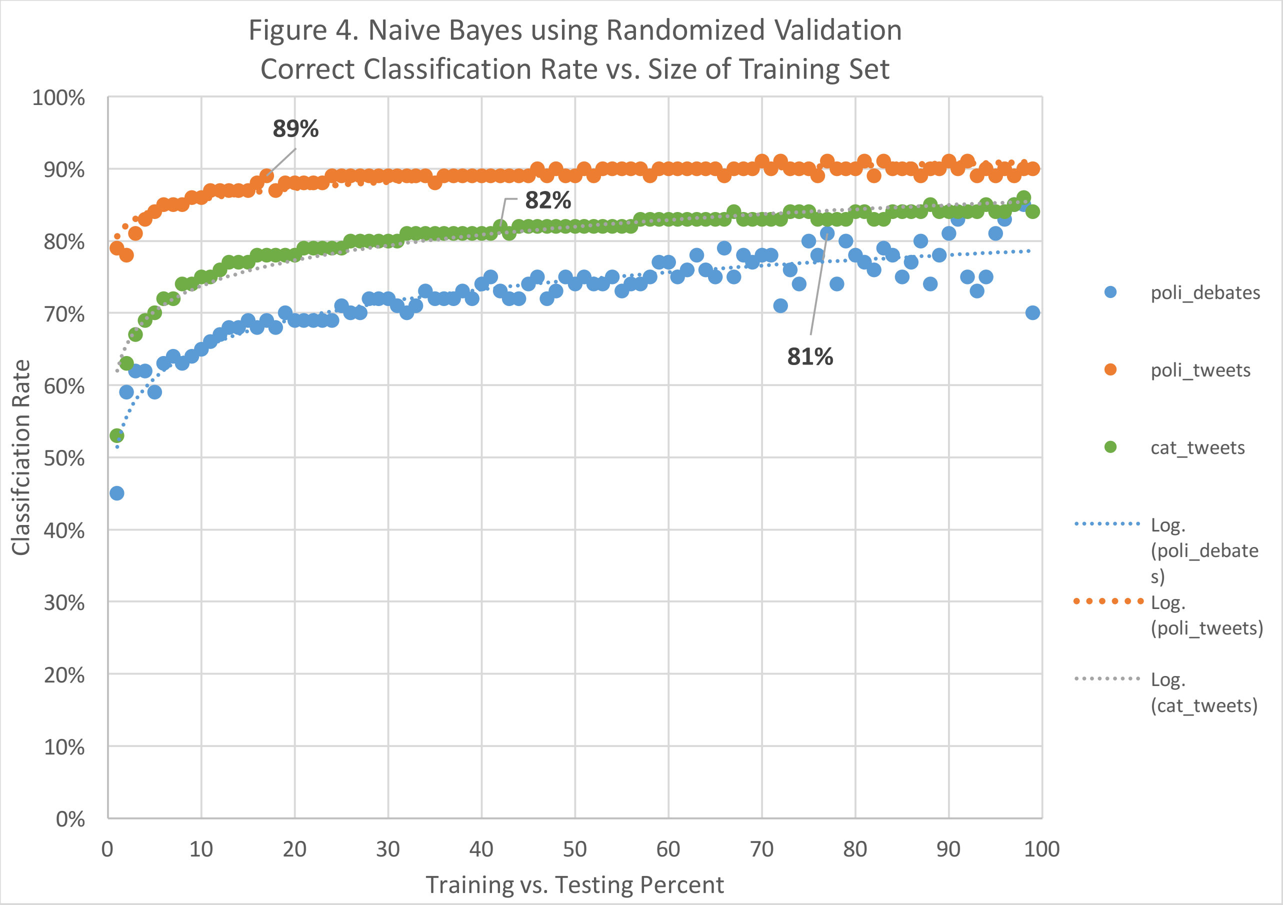


Figure 3. Naive Bayes using Randomized Validation
Correct Classification Rate vs. Size of Training Set



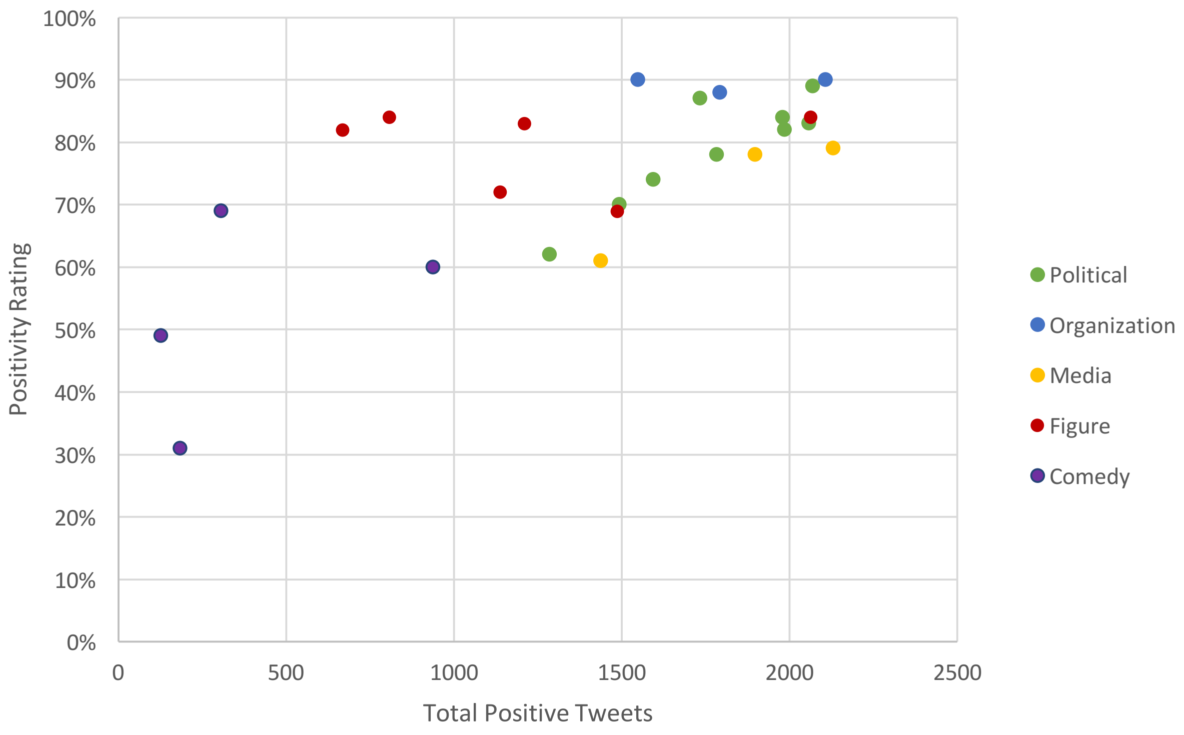


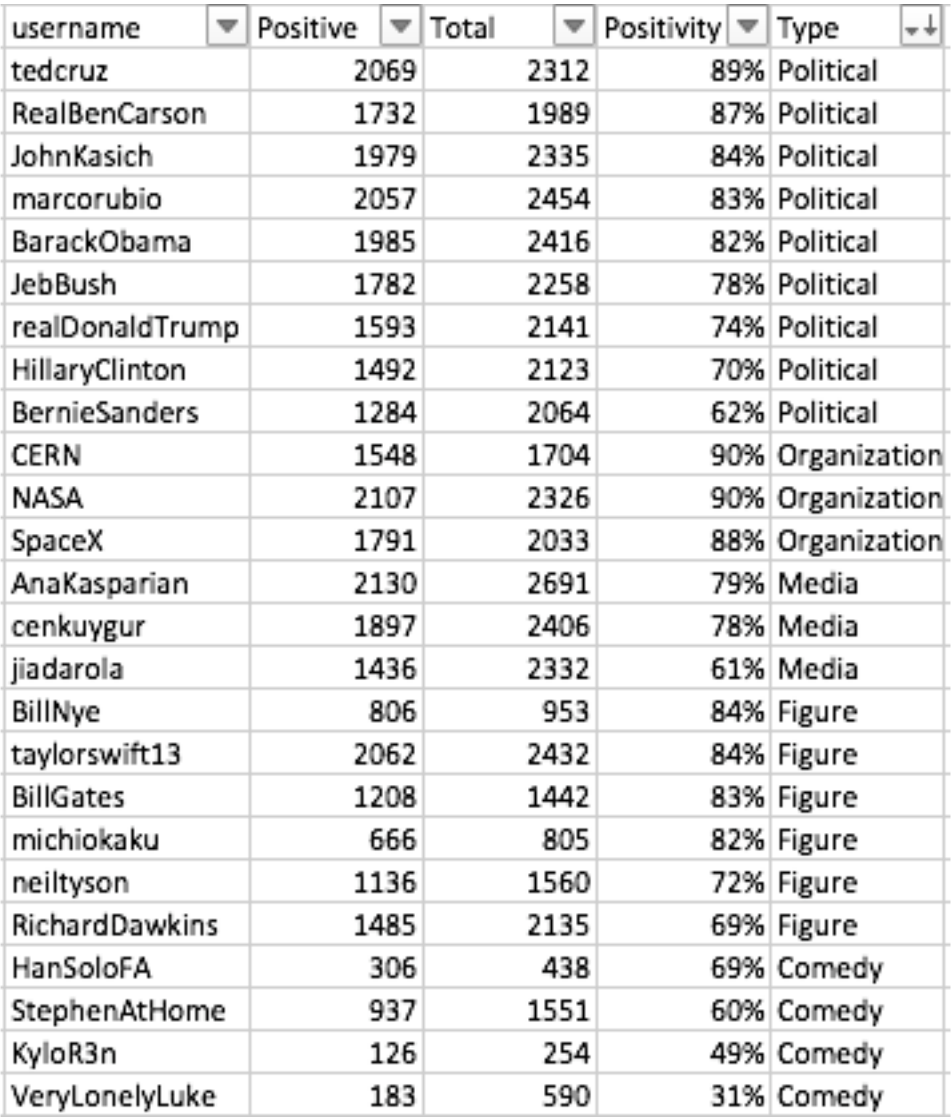
The randomized validation approach proved to be the most effective, as we observe in Figure 3. We found nearly 89% classification rate using a 78/22 training/test split for the forum-stemmed data. We explored a sentiment analysis case-study training our Naïve Bayes classifier using over 1.5 million tweets 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. We find similarly strong results in the additional analyses done with Figure 4, as we find 81% classification rate using a 77/23 training/test split for political debates, 82% classification rate using a 43/57 training/test split for categories of twitter users (sports, media, singers, science), and an 89% classification rate using a 17/83 training/test split for political affiliation tweets. We find better classification rates for twitter over debates potentially due to the larger sample sizes (2,000 vs. 60,000 and 20,000).

**Discussion**

In training our Naïve Bayes classifier using over 1.5 million tweets, we looked to use this classifier to make predictions about tweets outside the sample set. As such, we used the Twitter API to pull the last 3000 tweets (if they have that many) for around 30 users. The types of users were grouped into political, organization, media, figure, and comedy, as seen in Figure 5 and Table 1. Due to the unorthodox syntax of most tweets, we utilized libraries that addressed issues like retweets, emoticons, handles, hashtags, and links. Instead of just removing these types of language, we processed them in a way to better train our classifier. For instance, a smiley emoticon ☺, vs. a sad emoticon, ☹ can convey a positive or a negative sentiment.

Figure 5. Positivity Rating of Tweets Classified using Naive Bayes grouped by affiliation

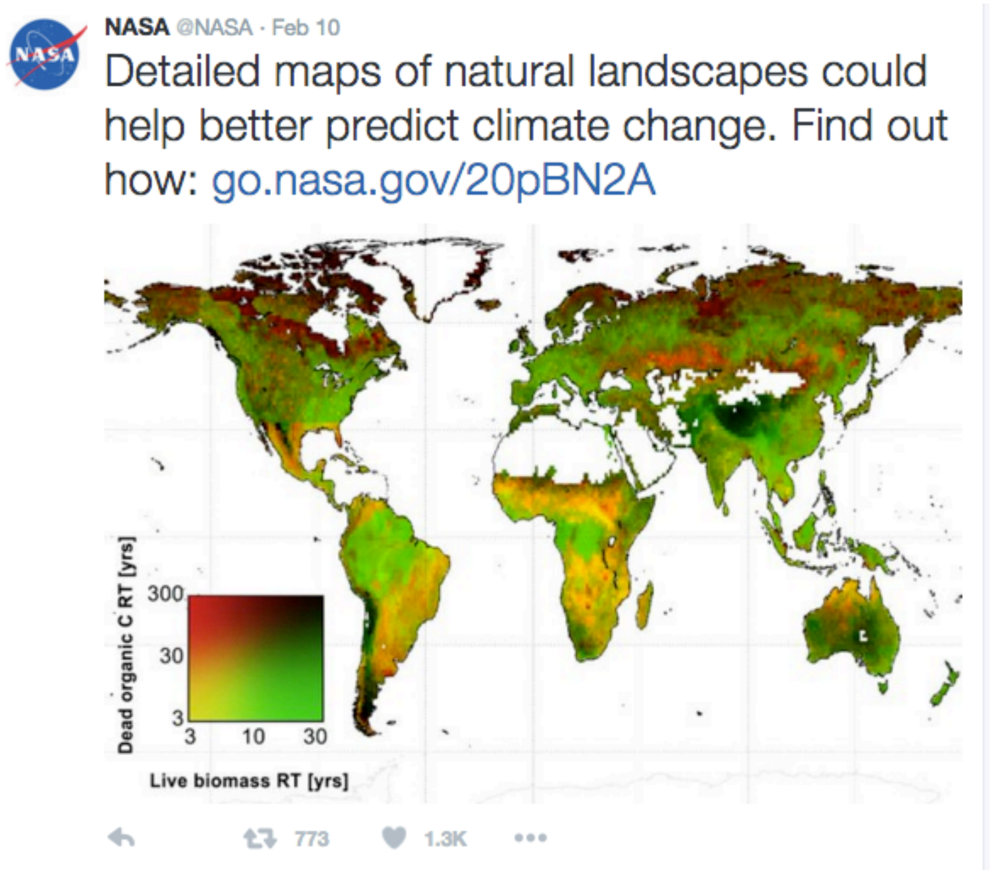




| username | Positive | Total | Positivity | Type |
|---|---|---|---|---|
| tedcruz | 2069 | 2312 | 89% | Political |
| RealBenCarson | 1732 | 1989 | 87% | Political |
| JohnKasich | 1979 | 2335 | 84% | Political |
| marcorubio | 2057 | 2454 | 83% | Political |
| BarackObama | 1985 | 2416 | 82% | Political |
| JebBush | 1782 | 2258 | 78% | Political |
| realDonaldTrump | 1593 | 2141 | 74% | Political |
| HillaryClinton | 1492 | 2123 | 70% | Political |
| BernieSanders | 1284 | 2064 | 62% | Political |
| CERN | 1548 | 1704 | 90% | Organization |
| NASA | 2107 | 2326 | 90% | Organization |
| SpaceX | 1791 | 2033 | 88% | Organization |
| AnaKasparian | 2130 | 2691 | 79% | Media |
| cenkuygur | 1897 | 2406 | 78% | Media |
| jiadarola | 1436 | 2332 | 61% | Media |
| BillNye | 806 | 953 | 84% | Figure |
| taylorswift13 | 2062 | 2432 | 84% | Figure |
| BillGates | 1208 | 1442 | 83% | Figure |
| michiokaku | 666 | 805 | 82% | Figure |
| neiltyson | 1136 | 1560 | 72% | Figure |
| RichardDawkins | 1485 | 2135 | 69% | Figure |
| HanSoloFA | 306 | 438 | 69% | Comedy |
| StephenAtHome | 937 | 1551 | 60% | Comedy |
| KyloR3n | 126 | 254 | 49% | Comedy |
| VeryLonelyLuke | 183 | 590 | 31% | Comedy |

**Table 1. Positivity results for Twitter validation set**

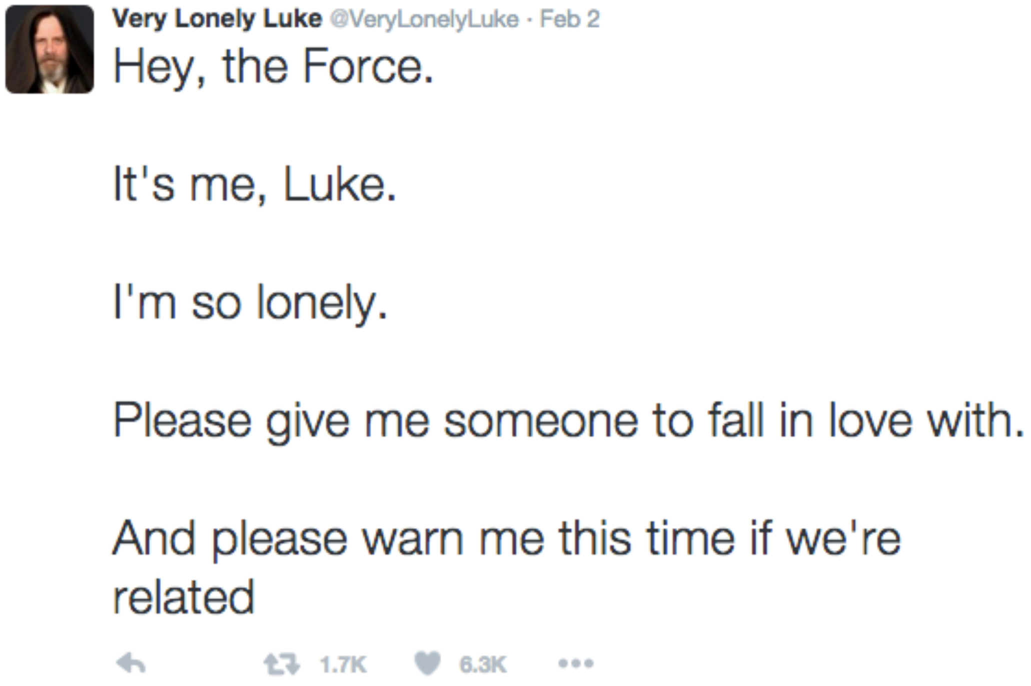
As we note from Table 1, all of the current presidential candidates as of early 2016 are included, as well as, scientific organizations and figures. A handful of media, celebrities, and comedy were pulled to round out the validation set. Some correlations we find from Figure 5 and Table 1 include more conservative politicians are more positive, scientific organizations are leading positivity on Twitter, secular media/scientists are less positive, and comedy/parody accounts are often more negative. We can compare the differences between positive and negative tweets in Figures 6 and 7.



NASA @NASA · Feb 10

Detailed maps of natural landscapes could help better predict climate change. Find out how: go.nasa.gov/20pBN2A

**Figure 6. Example of Positive Tweet.**



Very Lonely Luke @VeryLonelyLuke · Feb 2

Hey, the Force.

It's me, Luke.

I'm so lonely.

Please give me someone to fall in love with.

And please warn me this time if we're related

**Figure 7. Example of Negative Tweet.**

## 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 8 and 9 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.

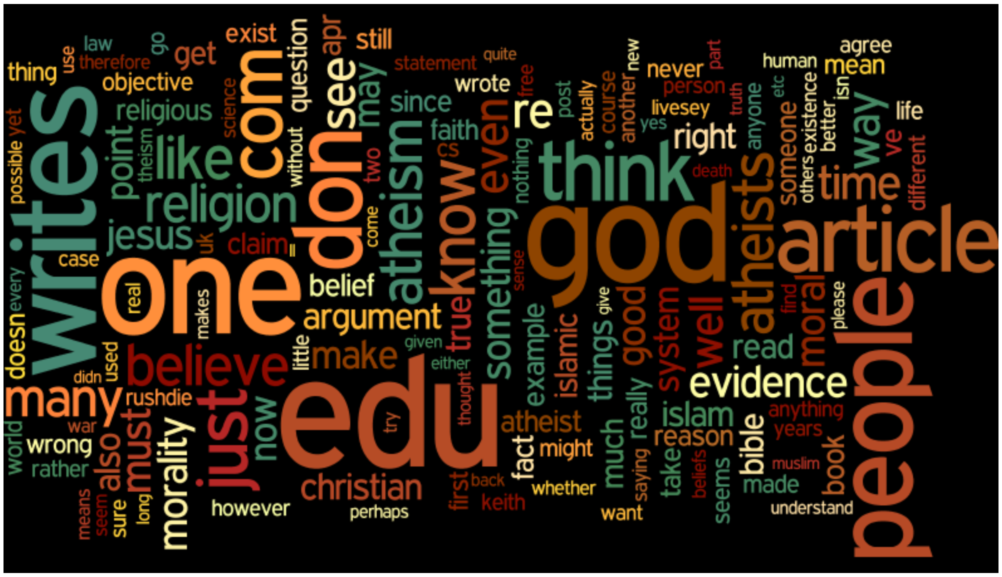


**Figure 8. Atheism Word Cloud**

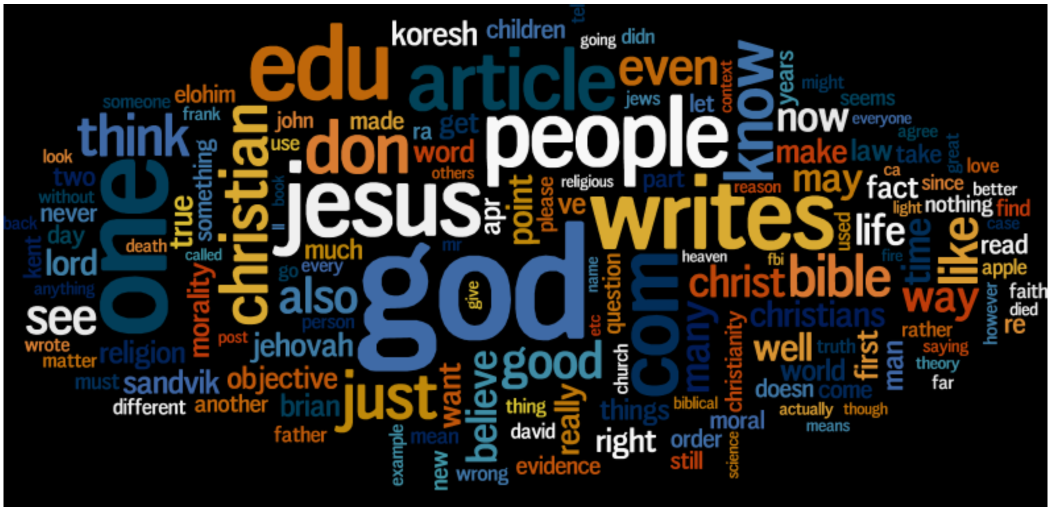


**Figure 9. Religion Word Cloud**

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 10.

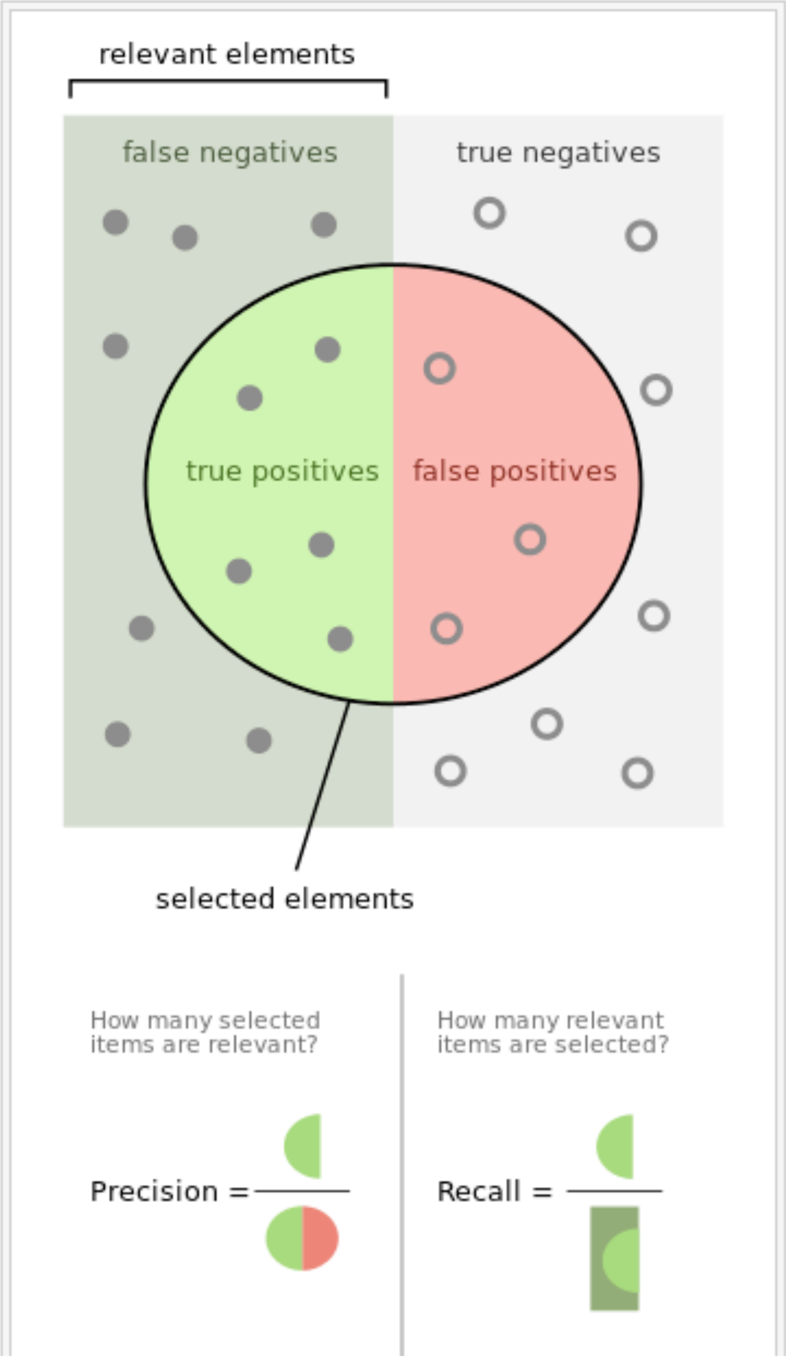
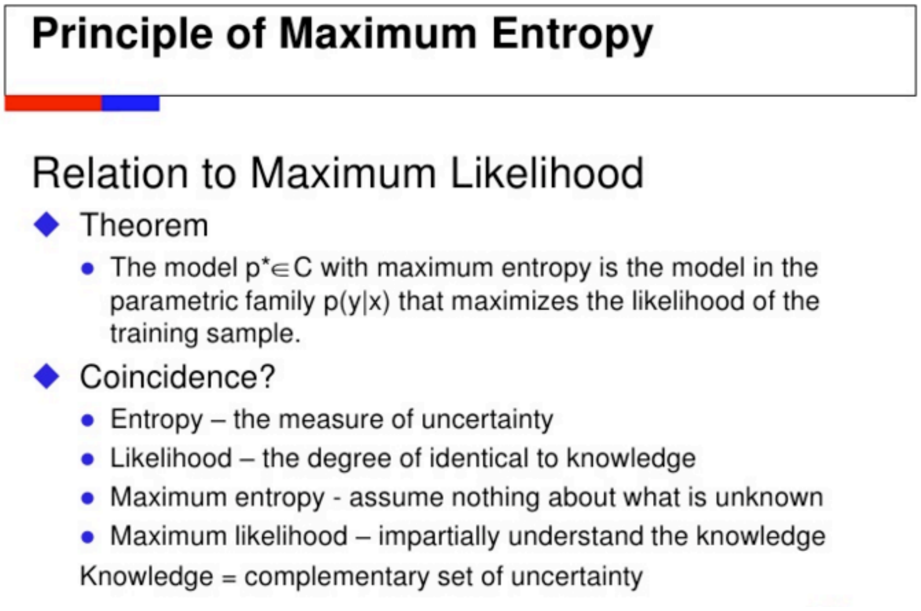


**Figure 10. Precision vs. Recall**

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 11 and 12.



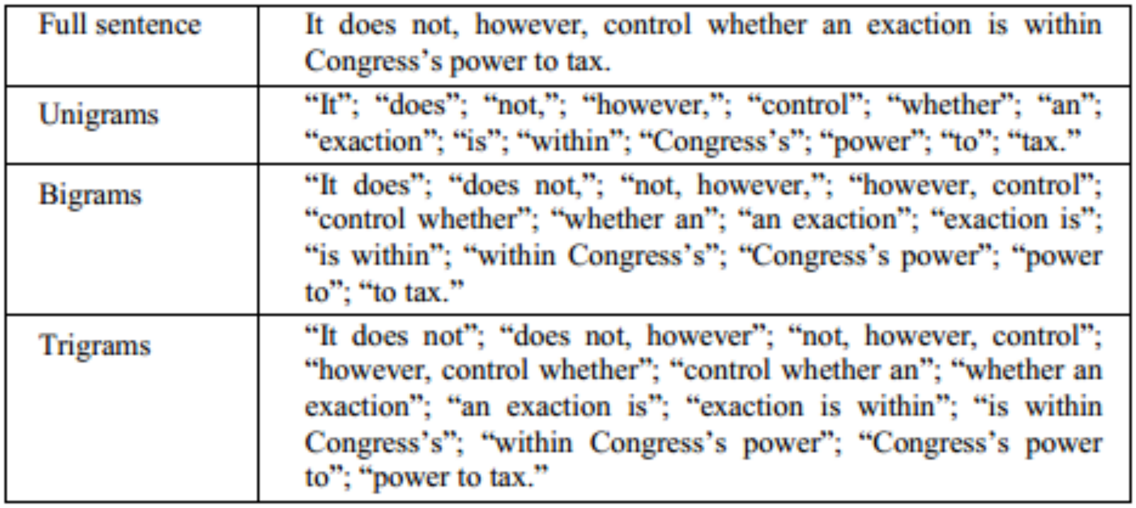
## Principle of Maximum Entropy

### Relation to Maximum Likelihood

- Theorem
  - The model $p^* \in C$ with maximum entropy is the model in the parametric family p(y|x) that maximizes the likelihood of the training sample.
- Coincidence?
  - Entropy – the measure of uncertainty
  - Likelihood – the degree of identical to knowledge
  - Maximum entropy - assume nothing about what is unknown
  - Maximum likelihood – impartially understand the knowledge

Knowledge = complementary set of uncertainty

**Figure 11. Further exploration of Maximum Entropy Classification**



| | |
|---|---|
| Full sentence | It does not, however, control whether an exaction is within Congress’s power to tax. |
| Unigrams | “It”; “does”; “not,”; “however,”; “control”; “whether”; “an”; “exaction”; “is”; “within”; “Congress’s”; “power”; “to”; “tax.” |
| Bigrams | “It does”; “does not,”; “not, however,”; “however, control”; “control whether”; “whether an”; “an exaction”; “exaction is”; “is within”; “within Congress’s”; “Congress’s power”; “power to”; “to tax.” |
| Trigrams | “It does not”; “does not, however”; “not, however, control”; “however, control whether”; “control whether an”; “whether an exaction”; “an exaction is”; “exaction is within”; “is within Congress’s”; “within Congress’s power”; “Congress’s power to”; “power to tax.” |

**Figure 12. 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)

**Usage**

*Quick implementation approach:*

Uploaded main Python program and one of the five datasets used (debates)

To run python program:

```
python documentClassifier_AT.py random
file_location num_iterations training/test_split
```

Example:

```
python documentClassifier_AT.py random
data/debateData/merged_debates.txt 5 90
```

For access to all work:

fork github repo https://github.com/adamtwig/AI_Project3.git

and contact Adam Terwilliger at terwilla@mail.gvsu.edu with any questions, comments, or concerns.