#Exposed: Detecting Twitter Trolls

Matthew Daniel and Cortlandt Bursey-Reece
CA: Amita Kamath

Motivation and Goals

- 48 million fake accounts (2017)
- 20% of 2016 election tweets were fake
- Goal: Need a way to filter these bots to protect democracy
- Goal: Classify troll and non-troll tweets

Data Collection

- We used a dataset of 95,000 politician (non-troll) tweets and 260,000 Russian troll tweets
- For each tweet, we stripped URLs, removed HTML, deleted punctuation, and discarded non-alphanumerics
- 70% of the politician and Russian tweets were used to train, 30% to test

Models

- **Naive Bayes:** Created feature vectors based on word extraction and character extraction

Features	Training Accuracy	Test Accuracy
Words	62.6%	61.4%
4-grams	80.7%	78.6%
5-grams	98.8%	93%

- **CNN:** Created a dictionary of word and n-gram frequencies after removing stop words. We then converted our dictionary into a word embedding model. We then implemented a Convolutional Neural Network using Keras. We filtered our embedding based on how many times a word appeared in our dictionary, and we ran our NN.

Method/Frequency	Training Accuracy	Test Accuracy	Loss
Words, 2	99.7%	64.3%	0.009
Words, 6	99.7%	63.2%	0.009
Words, 11	99.7%	60.5%	0.01
Words, 14	99.7%	63.4%	0.0091
6-grams, 2	80.4%	66.7%	0.434
10-grams, 2	77.8%	68.8%	0.483

Analysis and Discussion

- In Naive Bayes, our 'words' features performed worse than 4+-grams
- 'Words' CNN models had exceptional accuracy and loss, but it couldn't outperform n-grams in test-accuracy
- While our CNNs outperformed 'words' Naive Bayes, n-grams Naive Bayes had the best overall performance

Next Steps

- We could improve our CNN by using other word models. We would also run more experiments to find the variables that maximize our output
- While traditionally worse at text-classification, implementing an LSTM could yield interesting results