Christopher Lang

STA9792 – Natural Language Processing, Final Project

Baruch College, Fall 2017

## Introduction

In this project we wanted determine if it is possible to identify whether a news article is “real” or “fake” purely by its written content alone. We do not consider the source, or publisher, as we wanted to see if a more general model is possible. We don’t expect the model to be highly accurate, but the results can indicate whether this is feasible

## Data Source

We first get a source of label data, where articles are indicated to be real or fake, along with the article’s content such as title and the body of text. The two sources we looked at came from:

<http://www.opensources.co/>

<https://github.com/GeorgeMcIntire/fake_real_news_dataset>

We opted to only use the fake/real news dataset by George McIntire as it came with the articles title and body content. The open sources only provided URLs (at least from what I can tell). To make it more general, we favored a model “understand” the content of the article, not the source

The fake/real news dataset contained 6,335 news articles, with both the article’s title and body of text. We first created a TFIDF feature set from this corpus. Historically, and through my own experience, TFIDF features generally worked much better than simple term frequency or binary indicators. We theorize that TFIDF’s ability to “weigh up” unique words (or words with low document frequency) and significantly weight down heavily used words provided enough signal for many machine learning models to use for pattern detection

Other key choices made when developing the feature set include the removal of stop words and n-gram features. Stop words, being used very often to develop cohesive sentences but doesn’t contribute significantly to the text’s topic, were removed to reduce noise. Though TFIDF would’ve weighed it down, we found that due to their very high term frequency, they do not get weighed down enough

One of the biggest reasons to do the above steps is to reduce the number of features used to describe article content. If we combined all of the article’s title and body of text, we retrieved over 1.8 million features (unigram, bigram, combined) for only 6,335 articles. Modeling with this much features would be difficult (algorithms failing to converge, or distances almost always summing to nearly zero due to the sparse nature of the feature set). By cutting down on words and placing TFIDF filters, we can significantly reduce the number of features used

## Model Training

We decided to use a k-Nearest Neighbor (kNN) classifier on our TFIDF feature set. The kNN is a fairly simple model with minimal assumptions required. It will search for the nearest *k* articles based on shared features and uses a majority vote. We suspect that the model’s performance will not be optimal due to the varied words used. Collapsed features, such as topic modeling or word2vec, would help with this

To facilitate modeling, we only used the article’s title in this case. Article titles are often used to lure readers, and would be a good first step in determining whether the article or not is real or fake. It also has fewer words, hence fewer features to work with. The final added advantage is training time, as we planned to vary kNN’s parameters, requiring many models

To determine “good enough” parameters, many models and features were developed for the project. The follow model parameters were used:

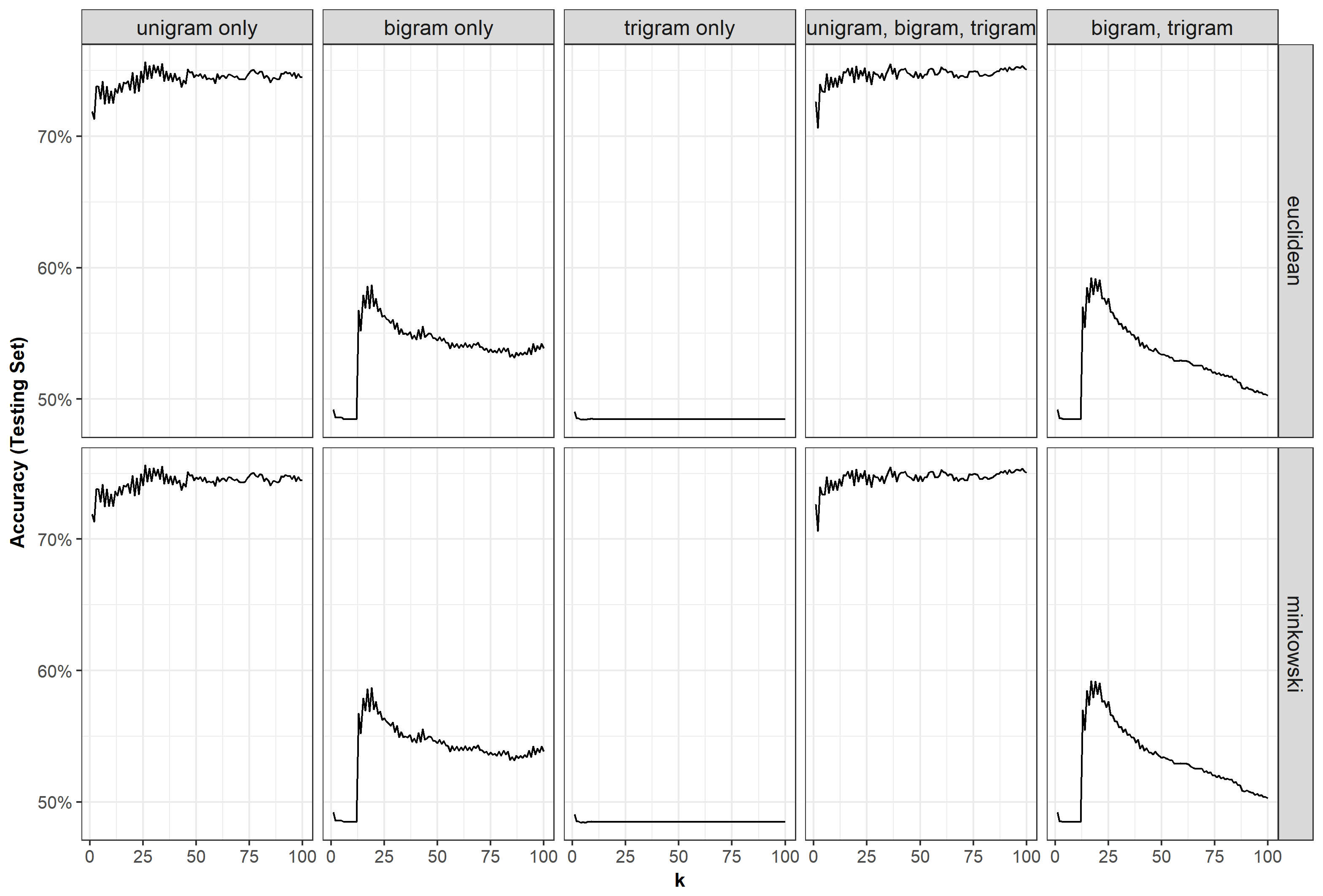
* The parameter *k* can vary from 1 to 100, in steps of 1
* The parameter *metric*, which was allowed to be either minkowski or Euclidean

We also allowed different n-grams to be used as features:

* Unigram only
* Bigram only
* Trigram only
* Unigram, bigram, and trigram
* Bigram and trigram

So in total, 1000 kNN models were develop and their result *k*, *metric*, n-gram feature, and accuracy was record, accuracy being the diagonal in the confusion matrix. Below outlines the testing accuracy of all the models

## Testing Results



Among these models, the best performing model was a kNN with k = 26, metric either minkowski or Euclidean, and a TFIDF unigram feature set. Below shows the validation set confusion matrix:

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Predicted | |
| REAL | FAKE |
| Actual | REAL | 232 | 83 |
| FAKE | 49 | 270 |

We see that the model did fairly well with real news (real news validation accuracy of ~83%) vs. fake news (fake news validation accuracy of ~76%). We suspect that though fake news use mostly similar wording as real news, a few words they’ll use often unique to their type of content will be enough of an indicator

## Conclusion

With a testing accuracy of approximately 75% and a validation accuracy of 79%, this particular model was somewhat acceptable. We figured performance would not be above 85% due to the significant number of features being generated. However, it is still promising, especially with a simple classifier like kNN. We expect performance would be better with a more refined feature set such as:

POS tagged unigrams – We expect this to provide richer information about the unigram themselves, though we also expect this to expand the feature set size

Term clustering – with any n-gram feature, we perform a cluster analysis to group them into larger entities, reducing the number of features. Being somewhat similar to topic modeling, this would’ve been a good dimensionality reduction with less information loss vs. simply filtering out low TFIDF n-grams

Latent Dirichlet Allocation, Latent Semantic Indexing, Principle Component Analysis – Proven and excellent ways to reduce the dimensionality of the dataset

Word2vec – A relatively new way of representing documents as vectors, performance could be improved using this feature extraction method, but performance improvements has been mixed from our experience

Ultimately, we believe our model could improve with better dimensionality reduction techniques that’ll preserve information in a smarter way than simple filtering