Fake News Detection

Exploring the Application of NLP Methods to Machine Identification of Misleading News Sources

Lauren Dyson & Alden Golab
CAPP 30255: Advanced Machine Learning for Public Policy
University of Chicago
Winter 2017

Abstract:

In this research summary, we explore the application of NLP techniques to the detection of 'fake news', that is, misleading news stories that come from non-reputable sources. Using a dataset obtained from Signal Media and a list of sources from OpenSources.co, we apply term frequency-inverse document frequency (TF-IDF) of bi-grams and probabilistic context free grammar (PCFG) detection to a corpus of about 11,000 articles. We find that TF-IDF of bi-grams fed into a Stochastic Gradient Descent model is capable of performing quite well at identifying non-credible sources, with PCFGs having slight effects on recall. We are skeptical about the generalizability of these findings, however, and include ample discussion on next steps for exploration in this space.

Introduction

In 2016, the prominence of disinformation within American political discourse was the subject of substantial attention, particularly following the surprise election of President Trump.[1] The term 'fake news' became common parlance for the issue, particularly to describe factually incorrect and misleading articles published mostly for the purpose of making money through pageviews. In this project, we seek to produce a model that can accurately predict the likelihood that a given article is fake news.

Facebook has been at the epicenter of much critique following media attention. They have already implemented a feature for users to flag fake news on the site;[2] however, it is clear from their public announcements that they are actively researching their ability to distinguish these articles in an automated way. Indeed, it is not an easy task. A given algorithm must be politically unbiased--since fake news exists on both ends of the spectrum--and also give equal balance to legitimate news sources on either ends of the spectrum. In addition, the question of legitimacy is a difficult one: what makes a news site 'legitimate'? Can this be determined in an objective way?

In this research summary we compare the performance of models using three distinct feature sets to understand what factors are most predictive of fake news: TFIDF using bi-gram frequency, syntactical structure frequency (probabilistic context free grammars, or PCFGs), and a combined feature union. In doing so, we follow the existing literature on deception detection through natural language processing, particularly the work of Feng, Banerjee, and Choi (2012) with deceptive social media reviews. We find that while bi-gram TFIDF yields predictive models that are highly effective at classifying articles from unreliable sources, the PCFG features do little to add to the models' efficacy. Instead, our findings suggest that, contrary to the work of Feng, Banerjee, and Choi's application, PCFGs do not provide meaningful variation for this particular classification task. This suggests important differences between deceptive reviews and so-called 'fake news'. We then suggest additional routes for work and analysis moving forward.

We have made all of our code publicly available. It can be found on github at: http://www.github.com/aldengolab/fake-news-detection.

---

[1] Articles are ubiquitous. For example: Caitlin Dewey, "Facebook fake-news writer: 'I think Donald Trump is in the White House because of me'", *Washington Post*, November 17, 2016, accessed 1/22/2017 at https://www.washingtonpost.com/news/the-intersect/wp/2016/11/17/facebook-fake-news-writer-i-think-donald-trump-is-in-the-white-house-because-of-me/?tid=sm_tw&utm_term=.cff9a5a0129d. See also, Sapna Maheshwari, "How Fake News Goes Viral: A Case Study," *The New York Times*, November 20, 2016, accessed 1/22/2017 at https://www.nytimes.com/2016/11/20/business/media/how-fake-news-spreads.html?_r=0.

[2] *Newsfeed FYI Blog*, December 12, 2016: http://newsroom.fb.com/news/2016/12/news-feed-fyi-addressing-hoaxes-and-fake-news/.

Background and Approach

Existing Literature

There exists a sizeable body of research on the topic of machine methods for deception detection, most of which has been focused on classifying online reviews and publicly available social media posts (Rubin, 2017). Particularly since late 2015 during the American Presidential election, the question of determining 'fake news' has also been the subject of particular attention within the literature.

Promising Approaches

Conroy, Rubin, and Chen (2015) outline several approaches that seem promising toward the aim of correctly classifying misleading articles. They note that simple content-related *n*-grams and shallow part-of-speech (POS) tagging have proven insufficient toward the classification task, often failing to account for important context information. Rather, these methods have been shown useful only in tandem with more complex methods of analysis.

Deep Syntax analysis using Probabilistic Context Free Grammars (PCFG) have been shown to be particularly valuable in combination with *n*-gram methods. Feng, Banerjee, and Choi (2012) are able to achieve 85%-91% accuracy in deception related classification tasks using online review corpora.

Other methods have included:

- Feng & Hirst (2013) implement a semantic analysis looking at *object:descriptor* pairs for contradictions with the text on top of Feng's initial deep syntax model for additional improvement.
- Rubin & Lukoianova (2014) analyze rhetorical structure using a vector space model with similar success.
- Sentiment and fact-based argument analysis (Pang & Lee, 2008).
- Language pattern similarity networks (Ciampaglia et al., 2015) requiring a pre-existing knowledge base.
- Social networks using inter-article links using centering resonance analysis (Papacharissi & Oliviera, 2012).

Ultimately, there is still much work to be done within the field to advance the work toward a well functioning model for detection.

Classification Models

Most classification models have been simple Naive Bayes or Support Vector Machine (SVM) approaches, which have worked substantially well. It is unclear the extent to which more complex modeling approaches have been tested for success. We implement the following models:

1. Support Vector Machines (SVM)
2. Stochastic Gradient Descent (SGD)
3. Gradient Boosting (GB)
4. Bounded Decision Trees (DT)
5. Random Forests (RF)

Our code relies upon the implementation of these methods in SciKit-Learn.[3] More information about these implementations can be found on their website.

Corpus Requirements

Rubin, Chen, and Conroy (2015) outline several requirements for a helpful corpus for use in these contexts (shortened for relevance):

1. Availability of both truthful and deceptive instances.
2. Verifiability of 'ground truth'.
3. Homogeneity in lengths.
4. Homogeneity in writing matter.
5. Predefined timeframe.
6. The manner of delivery (e.g. humor, sensational, newsworthy)

To deal with some of these challenges, we outsource some of corpus definitions to the website OpenSources.co[4] which compiles an ongoing list of fake and trusted news sources drawing from an academic at Merrimack College named Melissa Zimdars.[5] It is by no means perfect and has some detractors, as any list like this might. However, given the short time frame of the project, using a predefined list seemed prudent. Ultimately, our modeling approach should be data source independent and capable of using a better corpus or corpora when they are available.

Obtaining a corpus of news articles is notoriously difficult due to copyright issues. After many hours scouring the internet and reaching out to researchers at other Universities with little

[3] http://scikit-learn.org/stable/
[4] http://www.opensources.co/.
[5] http://www.merrimack.edu/live/profiles/586-melissa-mish-zimdars.

success, we found a dataset published by Signal Media in conjunction with the Recent Trends in News Information Retrieval 2016 conference to facilitate conducting research on news articles. The dataset contains about 1 million articles from a variety of news sources from September 2015. Sources include major news outlets like Reuters as well as local news sources and blogs. From this dataset, we filtered to include articles from verified reliable sources (labeled as 0) and verified unreliable sources (labeled as 1). We compiled our list of reliable sources and unreliable sources from OpenSources and FakeNewsWatch.

Our cleaned dataset contains 11051 articles. 3217 (29%) are labeled as fake. The reliable articles come from 14 unique sources. The unreliable articles come from 61 unique sources. In particular, for fake news our examples are heavily drawn from one source: Before It's News. This is not an ideal approach; we would like to have a more authoritative, vetted source for establishing reliable versus unreliable sources, and we would like our examples to be from a wider variety of sources.

Table 1: *Comparison of top unreliable and reliable source by article frequency*

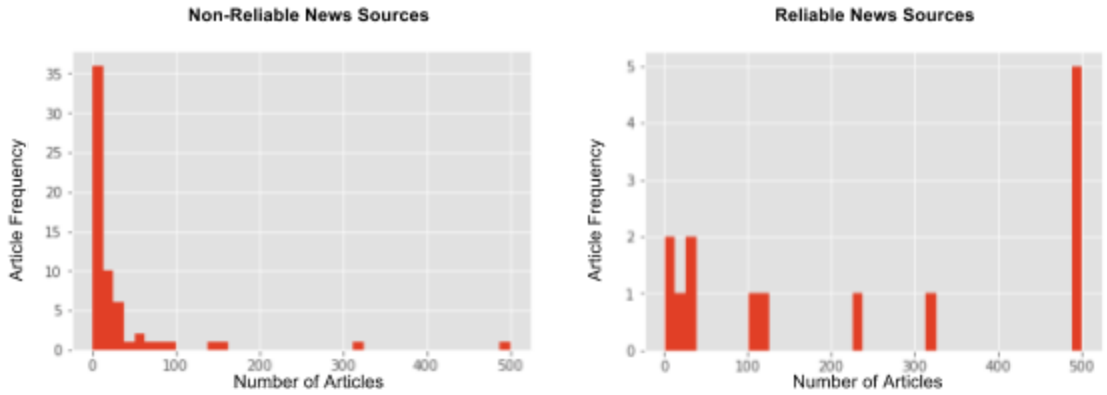
Top Five Unreliable News Sources		**Top Five Reliable News Sources**	
Before It's News	2066	Reuters	3898
Zero Hedge	149	BBC	830
Raw Story	90	USA Today	824
Washington Examiner	79	Washington Post	820
Infowars	67	CNN	595

Resampling to Account for Skewed Distributions

In order to limit the extent to which our models will primarily learn the difference between 'Before It's News' and *Reuters*, we force the distribution to cover a more limited range by randomly re-sampling the largest source contributors for a smaller *n*. You can see the final distribution of article frequency by source below in Figure 1.

We choose *n-max* of 500 articles for our implementation as it seemed prudent, though it is non-empirically based. We also do not drop low frequency sources in the interest of maintaining some heterogeneity of sources. The correct *n-max* (or a potential *n-min*) is an interesting research question in its own right. Additional avenues of research may consider varying this number to achieve an optimal result if facing similar corpus difficulties. We notice that before and after our re-sampling of the distributions we see a slight drop in Precision, indicating that we did fit to particular sources rather than the classes themselves with a more skewed distribution of sources.

***Figure 1**: Comparison of reliable and unreliable distribution by article frequency after resample*



Feature Generation

Our approach evaluates the performance of models trained on three feature sets:

1. Bigram Term Frequency-Inverse Document Frequency
2. Normalized frequency of parsed syntactical dependencies
3. A union of (1) and (2)

For feature generation, we rely on the relatively new Spacy Python package to conduct tokenization, part-of-speech tagging, syntactical parsing, and named entity recognition. Spacy is implemented in Cython (a superset of the Python language that allows C code for be generated from Python using the Python/C API), allowing for very fast performance compared to other NLP packages such as NLTK.

Several evaluations from peer-reviewed journals find that Spacy achieves performance on parsing and entity recognition tasks that is comparable to other widely-used tools, while having a significant advantage on speed.[6] This is why we chose to use Spacy over more established options such as the Java implementation of Stanford's Probabilistic Context Free Grammar.

From the raw article text, we use Spacy and SciKit Learn to generate the relevant features. We utilize Spacy's support for multi-threading to parallelize the feature generation process and

[6] E.g. see https://aclweb.org/anthology/P/P15/P15-1038.pdf
https://aclweb.org/anthology/W/W16/W16-2703.pdf

SciKit Learn's Pipeline feature to create fit-transform and transform methods that can be used on the training data and then applied to the test set.

Preprocessing

First, we scrub the articles of any mention of the name of the source. Because the reliable/unreliable classification is determined at the source level, this step is necessary to ensure the model doesn't just learn the mappings from known sources to labels. We also strip Twitter handles and email addresses (which often show up in journalist biographies) for the same reason.

Term Frequency-Inverse Document Frequency

The first feature set is vectorized bigram Term Frequency-Inverse Document Frequency. This is a weighted measure of how often a particular bigram phrase occurs in a document relative to how often the bigram phrase occurs across all documents in a corpus.

Because of the political nature of our corpus, we want to limit the model's knowledge of the people and institutions mentioned in the article text. Otherwise, we risk the model simply learning patterns such as "Clinton corrupt" which describe the topic and viewpoint of the text, rather than the outcome of interest (is this source reliable or not). Additionally, these patterns will be highly sensitive to the particular news cycle. To address this concern, we introduce a step during tokenization to use Spacy's named entity recognition to replace all mentions of named entities with a placeholder, e.g. <-NAME-> or <-ORG->.

We use SKLearn to calculate the TFIDF for each bigram within each document and build a sparse matrix of the resulting features. To keep the dimensionality of our data to a manageable size, we limit the vocabulary to only consider the top 3000 terms ordered by term frequency across the entire corpus. We did not experiment with different methods or thresholds for selecting the terms included in the vocabulary, or with different lengths of n-grams, but this may be an area to explore in future work.

Normalized Syntactical Dependency Frequency

We use Spacy to tokenize and parse syntactical dependencies of each document. Spacy's algorithm is a transition-based, greedy, dynamic oracle using Brown clusters that is comparable in accuracy to Stanford's PCFG but dramatically faster and more lightweight.[7]

[7] https://aclweb.org/anthology/P/P15/P15-1038.pdf

Each token is tagged with one of 46 possible syntactic dependency relations, such as “noun subject” or “preposition”. We count the frequency of occurrences of each dependency tag and normalize by the total number of dependencies in the document. Again, we use SKLearn to convert these frequencies into sparse matrices suitable for training models.

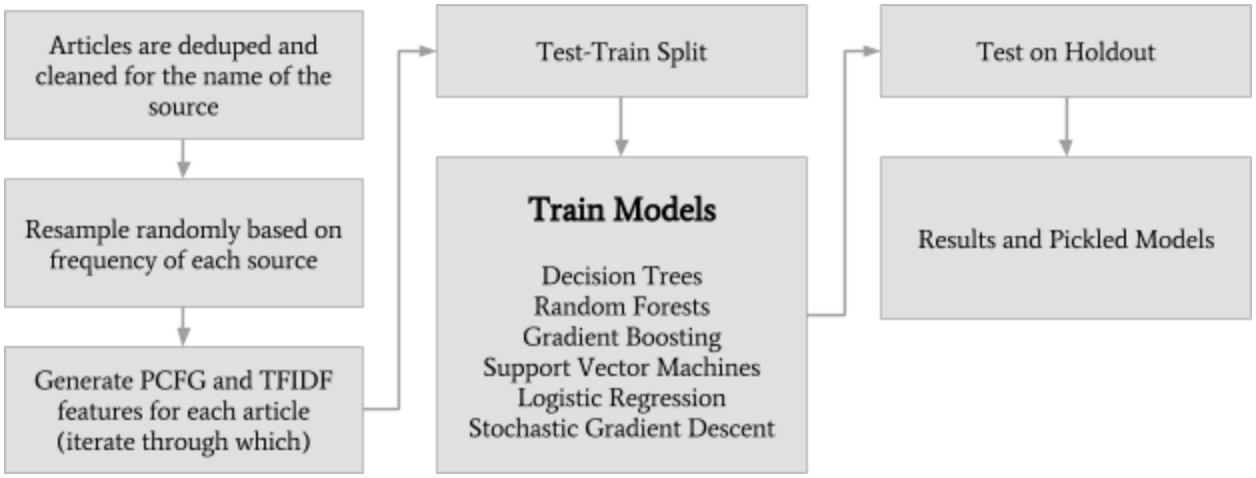
In total, we obtain 3000 features in TF-IDF family and 46 in the grammar family. Meaningful feature names are not currently available, limiting our ability to evaluate what specific characteristics of a document appear to be predictive of its legitimacy. As discussed below, this would allow us to better determine if the classifier is learning topical patterns or the outcome of interest and allow for a more thorough assessment of the generalizability of our results.

Modeling and Evaluation

Our Pipeline

After cleaning the data and generating features, we execute a 90/10 random test-train split on the dataset and feed it into a modeling pipeline. This pipeline iteratively fit models varying the tuning parameters with which they are executed up to 50 times, depending on the number of possible permutations for that model. These models are then tested on the 10% holdout data to understand their performance.

***Figure 2**: Pipeline representation*



Baseline Models for Comparison: Naive and Random

As a baseline comparison for understanding the performance of our models, we look to two methods. First, a naive model that predicts all majority class, in this case that all articles are from reliable news sources. Second, a model that randomly selects a classification for each article as either reliable or unreliable based on the posterior probability of that class in the

training set. These are the Naive and Random models, respectfully. We detail their performance in Table 2 below.

***Table 2**: Naive and random model performance across metrics.*

Model	Accuracy	Precision	Recall
Naive	67.89%	-	-
Random	56.41%	32.18%	32.18%

Combined Features Model Performance

Combining both feature sets, our models perform well above our baseline. Please see Table 3.

***Table 3:** Average model performance with both PCFG and TF-IDF bi-gram features at 0.7 score threshold for categorization.*

Model	AUC	Precision	Recall	Accuracy
Bounded Decision Trees	65.9% (.044)	66.9% (.062)	37.9% (.117)	67.6% (.028)
Gradient Boosting	75.6% (.122)	40.2% (.449)	16.1% (.223)	65.7% (.077)
Random Forests	80.0% (.092)	84.2% (.231)	18.4% (.176)	64.8% (.034)
Stochastic Grad. Desc.	87.5% (.003)	74.1% (.015)	71.7% (.035)	65.7% (.002)
Support Vector Mach.	84.3% (.012)	80.9% (.809)	44.5% (.063)	73.6% (.017)
Baseline	-	32.18%	32.18%	67.89%

Note: Standard Deviations are included within parentheticals.

We note that our best models tended to be Stochastic Gradient Descent (SGD) models, which, given that they tend to perform well with sparse and highly dimensional data, is not surprising. In particular, SGDs far outperform on precision *while retaining a high recall*, meaning that these models would work well both as identification of high priority articles in addition as 'fake news' filters.

TF-IDF Bigram Only Model Performance

Removing the PCFG features allows us to understand in more depth the value of those features in achieving these combined feature results. The results from this more limited feature run are displayed in Table 4, below.

Table 4: *Average model performance with only TF-IDF bi-gram features at 0.7 score threshold for categorization.*

Model	AUC	Precision	Recall	Accuracy
Bounded Decision Trees	60.7% (.061)	58.55% (.248)	23.3% (.179)	66.1% (.039)
Gradient Boosting	79.4% (.096)	41.0% (.439)	22.3% (.268)	68.7% (.080)
Random Forests	78.8% (.105)	82.9% (.182)	25.3% (.180)	67.6% (.033)
Stochastic Grad. Desc.	88.3% (.001)	88.8% (.028)	45.3% (.059)	77.2% (.012)
SVMs	85.6% (.012)	81.3% (.042)	48.1% (.058)	76.2% (.012)
Baseline	-	32.18%	32.18%	67.89%

Note: Standard Deviations are included within parentheticals.

Notice that the removal of PCFGs *improves* most of the metrics across our models. This is surprising, indicating that the PCFG features add little predictive value to the models. Indeed, the only noticeable decrease in performance is in our Recall figures for Decision Trees and SGDs.

PCFG Only Model Performance

Removing the TFIDF bi-gram features allow us to isolate the predictive value of PCFGs for this application in Table 5, below.

Table 5: *Average model performance with only PCFG features, classifying the top 5% of scores as positive (k = 0.05).*

Model	AUC	Precision	Recall	Accuracy
Bounded Decision Trees	50.0% (0)	40.9% (0)	10.8% (0)	60.1% (0)
Gradient Boosting	50.0% (0)	40.9% (0)	10.8% (0)	60.1% (0)
Random Forests	50.0% (0)	40.9% (0)	10.8% (0)	60.1% (0)
Stochastic Grad. Desc.	50.0% (0)	40.9% (0)	10.8% (0)	60.1% (0)
SVMs	50.0% (0)	40.9% (0)	10.8% (0)	60.1% (0)
Baseline	-	32.18%	32.18%	67.89%

Note: Standard Deviations are included within parentheticals.

Surprisingly, all of our models give the same result. Diving into the individual predictions, we find that all models produce the same rank order of scores. Note, also, that we've switched from a 0.70 threshold for classification to a top-k of 0.05. This is because the distribution of scores for these models have a particularly low mean with a tight range, such that determining an appropriate threshold for categorization was tedious and the 0.70 results weren't illuminating.

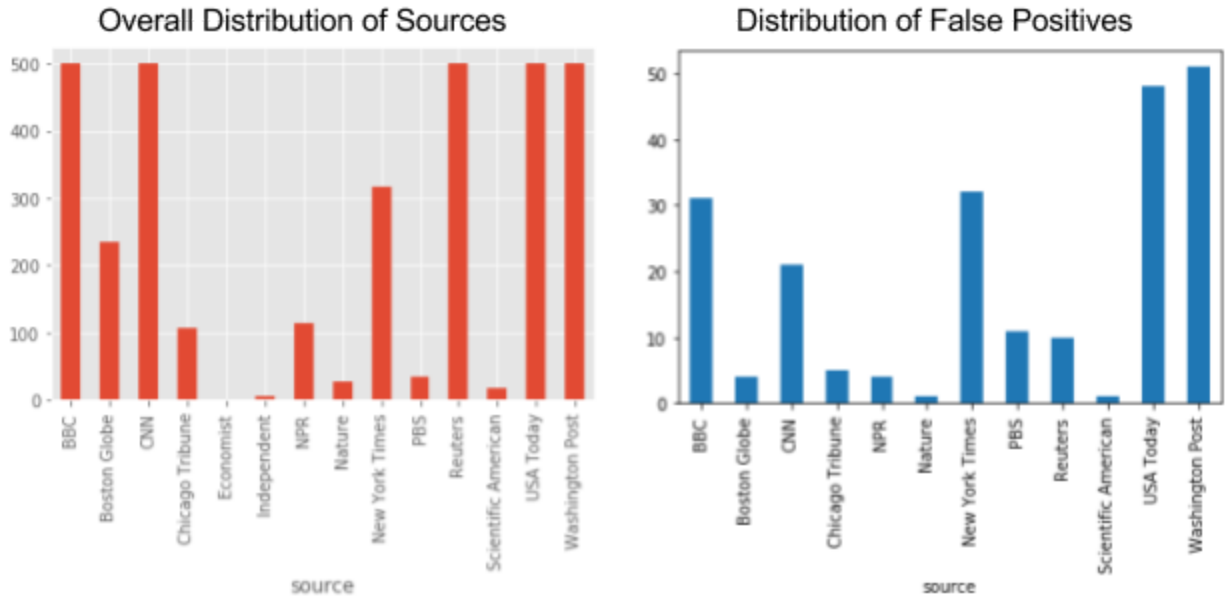
All this goes to indicate that in the case of this classification task, PCFGs do not add a strong source of information for classification on their own.

Where We Go Wrong

Exploring the results from our best performing models, SGDs, we notice some interesting results in terms of our incorrect predictions.

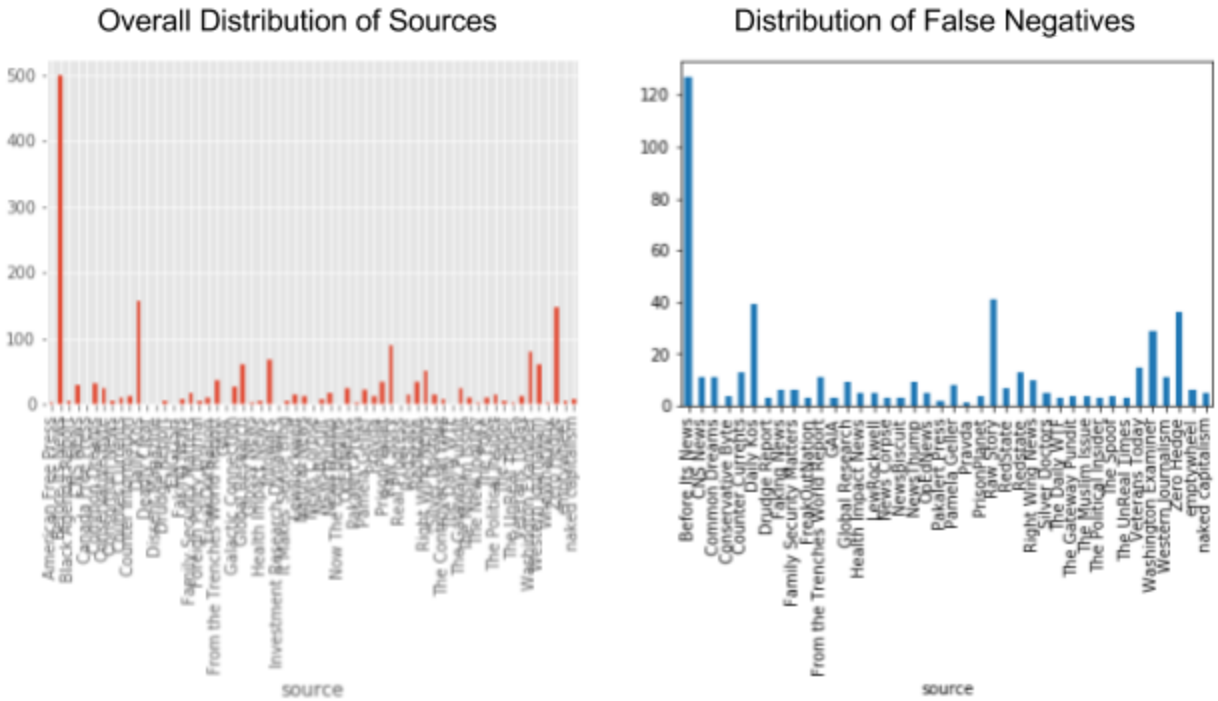
Looking at False Positives, we see that the distribution does not match the overall source distribution; instead, our false positives are concentrated within USA Today, Washington Post, and the NYTimes, for each of which we err in classification about 10% of the time. See figure 3 for more detail below.

Figure 3: *Comparison of distribution of false positives to overall frequency by article source*



Looking at False Negatives, it is more difficult to see the sources, since there are so many, however it is notable that the distributions, in comparison, look decidedly similar, with sources like ‘Before It’s News’, ‘Daily Kos’, and ‘Raw Story’ remaining modes in both charts.

Figure 4: *Comparison of distribution of false negatives to overall frequency by article source*



Given the difficulty of extracting feature names from our dataset, given technical limitations, it is difficult to know what accounts for the particular difficulties of correctly categorizing NYTimes,

USA Today, and Washington Post articles. Cursorily, there is no thematic trend within our false positives. This could indicate that some of the unreliable sites target these publications for similarity of language, diction, and prose, given their comparably larger readership and US base, but this would another site for further research.

Results and Next Steps

Results

Our discussion has touched on the following conclusions:

- Overall, the best performing models by overall ROC AUC are Stochastic Gradient Descent models trained on the TFIDF feature set only. SGD is known to work well for high-dimensional NLP tasks with sparse data.
- PCFGs do not add much predictive value, but balance the Recall for our top performing model. This indicates that PCFGs are good for a Fake-News Filter type implementation versus, say, targeting fake news sites for review.
- TFIDF shows promising potential predictive power, even when ignoring named entities, but we remain skeptical that this approach would be robust to changing news cycles. This would require a more complete corpus, however.

Limitations

There are substantial limits to our analysis that prevent broader generalizability.

- We evaluated our models using absolute probability thresholds, which may not be the most reliable for models where probability scoring is not well-calibrated. With more time, we would re-run evaluation using relative probabilities for labeling for better comparison.
- While TFIDF performs better, we are possibly overfitting to topics/terms important in this news cycle (November 2016). A vectorized approach like the one we pursued unfortunately makes it technically hard to see what individual features are most important, hampering our analysis.
- With limited verified sources of fake news in our dataset, we cannot claim that this approach would generalize to unseen sources.
- With a limited time period and heavy reliance on term frequency for classification, we cannot be confident that this approach would be consistently useful across news cycles.

This is, of course, in addition to the many source related limitations we explicated earlier in this summary.

Conclusions and Next Steps

Initial results show that term frequency is potentially predictive of fake news; an important first step toward using machine classification for identification. However, we remain concerned about overfitting and learning topical patterns that predict the partisan split of the legitimate vs. fake sources as identified by OpenSources.co. Further research should take seriously the challenge of finding a corpus that satisfies the conditions found on page 4 of this summary as a launching point for research. Ultimately an objective way of classifying 'fake' from legitimate news continues to be barrier that will make adoption difficult.

Additionally, the uninformative nature of PCFGs for this particular classification task indicates something potentially important about fake news versus other kinds of deception. Whereas fake reviews of restaurants might follow different syntactical structures to true reviews without intent, fake news is *intended* to mislead. In this case, it is likely that the unreliable sources will do their best to mimic the syntactical qualities of legitimate news sources. PCFGs' lack of information for prediction may indicate that other methods will need to be used in order to successfully carry out the classification task.

We suggest the following paths for further research:

- Curation of a more carefully selected article corpus.
- Focus on testing models across time--this will indicate whether the term frequencies are capable of processing unseen articles from new news cycles in addition to the training set.
- Layering on of additional techniques on top of term frequency and PCFGs.

Time Spent and Code Responsibilities

It is difficult to quantify precisely how much time was spent on this project. We would estimate conservatively approximately 30 hours per person over the course of the quarter.

Code Responsibility:

- Data Cleaning:
    - **Alden**: csv_transform.py
    - **Lauren**: get_articles.py
- Data Scraper:
    - **Alden:** all code
- Pipeline
    - **Alden:** model.py, model_loop.py, run.sh, run.py restrict_sources, some edits to transform_features.py
    - **Lauren:** transform_features.py, run.py, gen_features.py, edits to model_loop.py

Bibliography

1. Ciampaglia, G., Shiralkar, P., Rocha, L., Bollen, J. Menczer, F., & Flammini, A. 2015. “Computational fact checking from knowledge networks.” *PLoS ONE*. 10(6).
2. Conroy, Niall J. , Victoria L. Rubin, and Yimin Chen. 2015. “Automatic Deception Detection: Methods for Finding Fake News.” *ASIST '15 Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*. Article 82.
3. Feng, V. & Hirst, G. 2013. “Detecting deceptive opinion with profile compatibility.”
4. Rubin, V. & Lukoianova, T. 2014. “Truth and deception at the rhetorical structure level.” *Journal of the American Society for Information Science and Technology*. 66(5).
5. Pang, B. & Lee, L. 2008. “Opinion mining and sentiment analysis.” *Foundations and Trends in Information Retrieval*. 2(1-2): 1–135.
6. Papacharissi, Z. & Oliveira, M. 2012. “The Rhythms of News Storytelling on #Egypt.” *Journal of Communication*. 62: 266–282.
7. Rubin, Victoria L., Yimin Chen and Niall J. Conroy. 2015. “Deception Detection for News: Three Types of Fakes.” *ASIST '15 Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community*.