Fake News: Determining the Validity of Published Work

Simon Tao, Ryan Long, Juyon Lee
stao18@stanford.edu, rylong@stanford.edu, juyonlee@stanford.edu

Social Impact

Misinformation has been present in society since the dawn of human communication, but now in the **internet era, the reach of a single article has grown to 3.5 billion people.** The Internet is an extremely powerful channel for disseminating information; however, there is minimal to no regulation on the truth value of content that reaches the public. The burden of **determining source reliability is on the average consumer** of information, for whom it can be extremely difficult to discern data's validity. Overall, public access to the internet has allowed individuals to spread any message whatsoever — including fake news. Given that so much of America can be indoctrinated with a thought from a single fraudulent news source, there is a large need to provide a way to **accurately identify the validity of a news article.**

Data Collection

The dataset we will be using has 17,008 articles, consisting of 9,067 articles labeled as fake news and 7,941 articles labeled as real news. We have reserved 80% of our data as training data and 20% as our validation data.

1. **Cleaning Data:**

We split the articles into tokens delimited by white spaces, remove all punctuation from words, remove all words that are not purely comprised of alphabetical characters, and remove any word with a length of less than one character.

2. **Defining a Vocabulary:**

We used a bag-of-words model to identify fake news. Thus, it is important to define a vocabulary of known words before featurizing articles into their mathematical representations. This is especially critical in our case because we are using a large corpus of documents, meaning that the representation of non-predictive words will be high. We thus define a vocabulary using the *fit()* function in CountVectorizer to learn vocabulary across all of our training set documents.

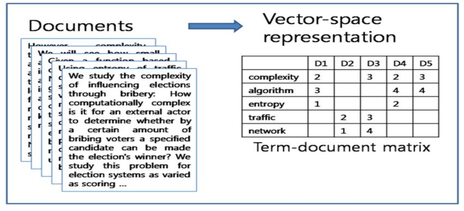
Challenges

Inherently, it is hard to identify the truth values in each of the articles solely based on vocabulary and rhetoric, but our algorithms hope to classify fake news based on observable patterns in word choice and style. One of the biggest challenges to executing our project was cleaning the data, as the different structures from the different classes made it harder to navigate around. Furthermore, identifying the best algorithm in terms of runtime and accuracy took a very long time to do, given our large data size. We also experienced issues with RAM limitations, but this was partially alleviated by running our code in Cloud.

Features

We decided on the bag-of-words strategy, where each article is transformed into a vector with each entry representing a score for a unique word in the sample document.

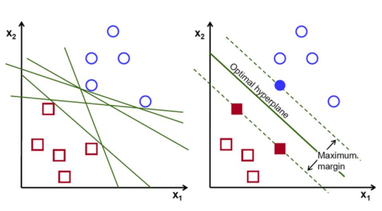
To score each entry in our data vectors, first, we convert each article into a frequency sparse vector using *transform()*. This gives each word a score equal to the amount of times it shows up in the document.



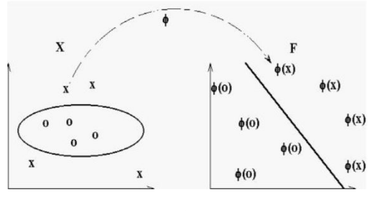
Then, we penalize words that are common across all of our sample documents and have low predictive value using a Term-Frequency Inverse Document Frequency Vectorizer (tfidfVectorizer). Words such as "a", 'the", etc. which serve only as syntactic glue but do not add semantic value will have their weights penalized as they are not essential to document classification.

Algorithms

We first tried using stochastic gradient descent with both linear and logistic separators and calculated the error by number of miscategorized articles. We then wanted to improve our classification accuracy by finding the most general linear separator, the maximum margin hyperplane, and thus ran the SVM algorithm on several kernels including RBF, polynomial, and sigmoid.



Maximum Margin

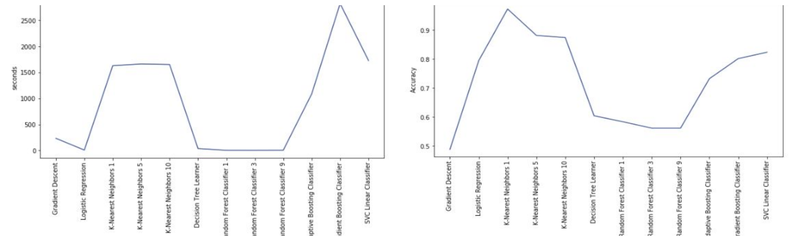


Kernel linear in Hi-D space

Finding the polynomial kernel to be most accurate, we tried non-linear classifiers like k-nearest neighbors and decision tree classifiers. After testing both the k-nearest neighbors and random forest classifier on a toy set, we were able to determine that the optimal k = 10 and the optimal n = 1. We then improved the tree classifier algorithms with both adaptive and gradient boosting.

Experimental Procedure and Results

Classification Algorithm	Accuracy	Run Time (sec)
Linear Discriminant	0.4887	233.66
Logistic Regression	0.7953	8.46
SVM with Linear Kernel	0.824	1726.66
SVM with Polynomial Kernel	0.842	3540.412
K-Nearest Neighbors K = 10	0.875	1651.72
Decision Tree Learner	0.605	37.55
Random Forest Classifier n = 1	0.584	3.887018204
Adaptive Boosting on Extra Trees Classifier	0.733	1085.74
Gradient Boosting on Extra Trees Classifier	0.802	2820.83



Error Analysis

Upon analysis, we found that the K-Nearest Neighbors algorithm with K = 10 was the most accurate since groups with similar words and patterns found throughout articles are assigned similar scores. We then looked at articles that the K-Nearest Neighbors misidentified to perhaps add features that would allow it to better classify articles. The misclassified articles were generally on topics that included more extreme words like "radical", "illegals", "attitudes" and "opinions". It is difficult for our classifier to identify the difference between a testimonial and facts.

Future Work

One obstacle in developing machine learning algorithms for identifying fake news is the lack of comprehensive benchmark datasets. However, with larger data sets it becomes increasingly important to be able to store such large data which we could do by hashing. Also, implementing a deep neural network to help reduce the features over time would speed up runtime. Additionally, this algorithm is only able to identify fake news by the style in which it is written and does not actually check the truth value in the statement. An improved algorithm, given a dataset of facts could empirically determine whether the statement is factually accurate.