A comprehensive analysis of vector space models for fake news classification

Abhinav Choudhury
Department of Computer Science
North Carolina State University

# Completed as an independent study project as part of CSC 630, Spring 2017 under the supervision of Prof. Min Chi, Department of Computer Science, NC State University

Introduction

Fake news articles disseminated via social media like Facebook and Twitter has been a major problem on the internet for years. The days leading up to the 2016 US elections saw a proliferation of fake and politically biased news articles with many claiming that the election results were being affected by these. This evidently demonstrates the power of influence of news spread through social media and mandates that a fact-checking system or a filter be in place to flag any news article of questionable content. Traditional methods make use of human fact checking, such as the PolitiFact website run by the Tampa Bay Times, which relies on its editors to manually fact check political information circulated on the internet. In this report, we explore different vector space representations of text articles and their performance on classifying fake news articles from real ones. We also detail the characteristics of a robust dataset consisting of fake and real news articles that we have built for training purposes.

Types of Fake News

Not all new misleading news articles are created with the same purpose. The following are the major categories

1. ***Fake news or hoax***: Websites that post articles with the purpose of intentionally deceiving readers, possibly mimicking existing real news websites. e.g. RealNewsRightNow.com, ABCnews.com.co
2. ***Satire/parody***: Websites intended to be funny with no direct intention of deceiving readers but may still lead to spread of misinformation. e.g. TheOnion.com,
3. ***Bias/conspiracy theories***: Websites consisting of a mix of biased opinions and made up conspiracy theories.

We speculate that the content in these websites differ distinctively from genuine news sources in their style of writing and choice of words. We believe that a machine learning model should be able to accurately model this difference based solely on the text of the articles. For the purpose of this paper, we have classified news articles belonging to any of the 3 categories above as “Fake” since our primary goal is only to red-flag an article if it is not deemed genuine and truthful.

The Data

In order to assess the performance/reliability of a machine learning system in identifying fake news, hoaxes and satire from genuine news articles, a robust dataset of news articles is a prerequisite and building this dataset is no trivial task. We expect that the data to satisfy the following requirements:

1. **Topic diversity**: Having news articles from a plethora of categories such as politics, business, sports, entertainment, etc increases the generalization capability of our trained model. We try to maintain a balanced topic distribution as far as possible but this is not easy due to the political bias of most fake news sites.
2. **Source diversity**: Different news websites and sources have different styles of writing, preference of word usage, etc. To factor in such differences in our model, articles need to be scraped from a variety of different sources.
3. **Good class distribution**: The class distribution should ideally be balanced, i.e. roughly equal number of instances for Fake and Real classes.

For articles categorized as fake news, hoaxes and satire, the data was gathered by scraping the following websites:

1. www.theonion.com (Satire)
2. www.politicops.com (Fake/Hoax/Bias)
3. www.realnewsrightnow.com (Fake/Hoax/Bias)
4. www.enduringvision.com (Fake/Hoax/Bias)
5. www.civictribune.com (Fake/Hoax/Bias)
6. www.newsbiscuit.com (Fake/Hoax/Bias)

A total of 3313 articles were gathered from these websites which have been assigned the class "Fake" regardless of the category being satire, fake, hoax or bias. Note that since most of these websites focus largely on political content, the topic of the articles collected might be biased towards the politics category.

In addition, an openly available fake news dataset on Kaggle[2] was used, which had 12403 articles in English belonging to the time period October 2016 and November 2016, and categorized as being fake, unreliable, untrustworthy or junk. We added these articles to our dataset, coalescing them under the "Fake" class. This resulted in a total of **15716** fake news articles.

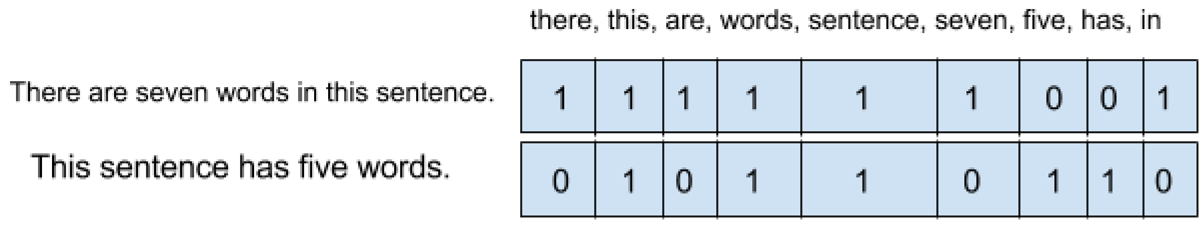
For authentic news, we used a collection of articles from **The New York times** and **The Guardian** since these are the popular and reliable news sources, and also have active developer APIs for mining news articles. A publicly available dataset of news articles from **BBC news**[3] was also appended to the list of real articles, adding up to a total of **12591** real news articles.

Vector space models for text

The next phase involves converting the text articles to a numerical fixed-length vector representation interpretable by machine learning models. The following techniques are frequently used for vector space models:

1. Bag-of-words (BoW): The bag-of-words model is a simple and intuitive model, but works surprisingly well for methods involving calculation of a similarity measure between documents. It is commonly used in document clustering tasks. The idea behind the model is to generate a vocabulary based on the unique words collected from documents (or articles) in the corpus, and then projecting each article/document into a vectorized representation by counting the

occurrence of each word of the vocabulary in the document/article. Either the entire vocabulary or a subset of most commonly occurring words can be used.



	there	this	are	words	sentence	seven	five	has	in
There are seven words in this sentence.	1	1	1	1	1	1	0	0	1
This sentence has five words.	0	1	0	1	1	0	1	1	0

Vocabulary: {There, this, are, words, sentence, seven, five, has, in}

A major flaw of the bag-of-words model is that it does not account for the ordering of words in a document, and as such any random permutation of words of a document would end up having the same representation. For example, the following sentences, although different semantically, would have the same representation under the BoW model:

The quick brown fox jumps over the lazy dog
The quick dog jumps over the lazy brown fox

Another issue with the BoW model is its inability to put adequate weights on important, representative words in the corpus vs frequently occurring words in the language like prepositions and pronouns. For example, using the BoW model, words like "the", "a", "an" which occur very frequently in English text would always end up with high counts.

2. Term-frequency Inverse document frequency (tfidf): tfidf combines the frequency of occurrence of each term with the inverse of its document frequency (the number of documents in the corpus it shows up in). This dampens the high frequency scores of words that appear frequently in all documents such as pronouns and prepositions ("this", "an", "but", etc)
Given a corpus of documents D, the tfidf score for a term *t* appearing in document *d* is defined as:

$$tfidf(t, d, D) = tf(t, d).idf(t, D)$$

where $tf(t, d)$ is the number of times term *t* appears in document *d,* and $idf(t, D)$ is the logarithmically scaled inverse of the fraction of documents containing t.
A high weight in tf–idf is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents

3. N-gram models: N-grams models solve the problem of incorporating the ordering of words into vector representations. It extends the BoW and tfidf models by using not only individual words to form the vector representation but also groups of N continuous words extracted from the corpus of documents. For instance, given the corpus of sentences we used to illustrate the Bag-of-words model

   There are seven words in this sentence.
   This sentence has five words.

   a 2-gram model would extract the following 9 groups as part of the vocabulary other than the 9 unique words seen in the corpus:

   (there are), (are seven), (seven words), (words in), (in this), (this sentence),
   (this sentence), (has five), (five words)

   This creates a vector representation of size 18 for each sentence (or document).

One issue with the N-grams models is that the term-document matrix created is extremely sparse and the sparsity of the increases with N.

4. Latent semantic analysis: Latent semantic analysis, also known as latent semantic indexing, solves the problem of synonymy and polysemy in language vocabularies. In other words, the vocabulary represented by a vector space model usually contain words that are similar in meaning (synonymy) or contain words that have multiple interpretations (polysemy). Latent semantic analysis takes the term-document matrix (from tfidf or bag of words model) and transforms it into a lower rank matrix, preserving the rows (denoting the documents) while reducing the number of columns (the number of topics) by compressing words with similar meanings into a single topic.

Experiments

We experimented with vectorization of the text using each of the vector space models detailed above and for 3 different vector lengths of 250, 500 and 1000 respectively. These vectors were then fed into a gradient boosted decision tree (GBDT) classifier based on the LightGBM[6] library in Python. GBDTs have recently emerged as the best out-of-the-box classifier that perform incredibly well in most practical problems and have been used in winning entries in many Kaggle competitions.

The gradient boosted decision tree classifier in the LightGBM model takes the following parameters which control the learning capacity and the generalization capability of the model:

- Maximum depth of trees (max_depth): Deeper trees tend to overfit
- Number of estimators (n_estimators): Number of trees to build
- Early stopping iterations (early_stopping): Stop if error metric has not improved in this many iterations
- Column sample (col_sample): Random fractional subset of columns to consider for building each tree
- Number of leaves (n_leaves):

We ran cross-validation tests with the following parameter combinations:

```
max_depth = {4, 5, 6, 7}
n_estimators = {500, 1000, 2000, 3000}
Early_stopping = {5, 10, 20}
col_sample = {0.5, 0.75, 1}
N_leaves = { 25, 50, 100 } [Only for the TFIDF and BoW experiments ]
```

This resulted in a total of 432 (4 x 4 x 3 x 3 x 3) parameter combinations. For each parameter combination, we ran cross-validation consisting of 4 iterations over which the results were averaged. Each iteration of cross-validation did a stratified shuffle split of the data into training and validation sets in a 75:25 ratio.

Results

We list below the performance of the best model and the associated parameters of the model for each vectorization method used and for each of the 3 different vector sizes for each method. Performance metrics are over the validation set and averaged over 4 cross-validation iterations. Model parameters are in order: (max_depth, n_estimators, n_leaves, early_stopping, col_sample).

TFIDF

Vector size	Model Parameters	Binary Logloss	Accuracy	Precision	Recall	F-measure
1000	(5, 3000, 25, 20, 0.5)	0.078308	**0.971422**	0.985794	0.962395	0.973949
500	(6, 3000, 100, 10, 0.5	0.08891	0.965805	0.983794	0.954123	0.968731
250	(7, 2000, 100, 5, 0.5)	0.102938	0.960753	0.982185	0.946488	0.963991

Bag-of-words

Vector size	Model Parameters	Binary Logloss	Accuracy	Precision	Recall	F-measure
1000	(7, 3000, 25, 5, 0.5)	0.35152	**0.860357**	0.82689	0.946679	0.882731
500	(7, 3000, 100, 10, 1)	0.364877	0.853893	0.82182	0.940825	0.877299
250	(7, 3000, 100, 10, 1)	0.387485	0.842377	0.810808	0.93408	0.868075

Latent semantic analysis

Latent semantic analysis was run after extracting TFIDF representations of size 10000. Note that for the tests below, num_leaves was removed as a variable factor since there was not much difference noticeable for differences in values of num_leaves. The value for num_leaves was fixed at 50.

Vector size	Model Parameters	Binary Logloss	Accuracy	Precision	Recall	F-measure
1000	(6, 3000, 50, 20, 0.75)	0.070697	0.972517	**0.98919**	0.960995	0.974887
500	(5, 2000, 50, 5, 0.75)	0.07094	0.973082	0.987737	0.963477	0.975455
250	(6, 3000, 50, 10, 0.5)	**0.067607**	**0.97446**	0.98708	**0.966658**	**0.976759**

Conclusion

The results using the TFIDF vectorizer indicate that an extremely robust classifier (**best accuracy 97.14%**) can be built to distinguish fake news articles from real ones based only on the text content of the articles, and one that is agnostic to semantics and word order. The robustness of the classifier is indicated by the fact that all performance metrics typically have high values (**96-98%**). Binary logloss, or binary cross-entropy, scores are also low (< **0.1**), indicative of a good probabilistic classification. The experiments also seem to indicate that representations using larger vector sizes also result in better performing classifiers.

Unsurprisingly, for the parameter combinations we experimented with, the TFIDF features outperform the Bag-of-words vector model and the 2-grams vector model. The results for the 2-grams models are not published here.

Classifiers based on the features extracted using latent semantic indexing perform marginally better (**best accuracy 97.44%**) than those based on TFIDF features. This is expected as LSI uses larger TFIDF representations and thus has more information content. However, the extra computational effort and time of performing LSA over using base TFIDF features may not be worth the effort.

References

[1] Clark, Stephen. 2013b. Vector space models of lexical meaning. In S. Lappin and C. Fox, eds., Handbook of Contemporary Semantics, 2nd ed.. Malden, MA: Blackwell. In press; https://www.cl.cam.ac.uk/~sc609/pubs/sem_ handbook.pdf
[2] Kaggle fake news dataset, https://www.kaggle.com/mrisdal/fake-news
[3] BBC News datasets, http://mlg.ucd.ie/datasets/bbc.html
[4] Eugenio Tacchini, Gabriele Ballarin, Marco L. Della Vedova, Stefano Moret, Luca de Alfaro, Some Like it Hoax: Automated Fake News Detection in Social Networks
[5] Victoria L. Rubin, Niall J. Conroy, Yimin Chen, and Sarah Cornwell, Fake News or Truth? Using Satirical Cues to Detect Potentially Misleading News
[6] Conroy, Rubin and Chen, Automatic Deception Detection: Methods for Finding Fake News
[7] LightGBM: A fast, distributed, high performance gradient boosting (GBDT, GBRT, GBM or MART) framework, https://github.com/Microsoft/LightGBM