Machine Learning Approaches to Capture the Reliability of News Articles

Rares Buhai and Tristan Thrush
{rbuhai, tristant}@mit.edu

Abstract

Following the United States presidential election of 2016, when it was claimed that fabricated stories distributed on social media influenced the results, fake news has become a global topic of discussion; it is clear that a solution to the fake news pandemic is needed. While many platforms are developing methods of filtering such unreliable content, most of them rely on human fact-checking. We are unaware of any automatic fake news filtering deployed at large scale. We present here multiple machine learning models that aim to capture the reliability of news articles. Specifically, we trained classifiers to distinguish reliable articles from unreliable ones, and we trained a model capable of generating articles in both categories.

1. Introduction

In order to contribute to solving the fake news pandemic, our research goal is to model the expected reliability of an article instead of its factual accuracy. This is because a perfect fake news classifier would need to evaluate truth claims, which is above the capabilities of artificial intelligence algorithms today. We define an unreliable article as one that the reader should distrust until confirmed from multiple source. In addition to seemingly fake articles, we also categorize extremely biased and hateful articles as unreliable.

This paper explores several machine learning models, to characterize the reliability of news articles. In particular, we provide several models that classify news as unreliable or reliable. We conducted experiments with an SVM, a basic feed-forward neural architecture, and finally an LSTM architecture. The resulting classification accuracies are over 75% when our models are trained on news titles and over 90% when they are trained on full articles. We also examine the different characteristics of unreliable and reliable news articles by providing an LSTM that is trained on such articles and generates news snippets.

2. Data Collection

To construct our dataset, we scraped news articles from the Internet. We identified reliable and unreliable sources, and then we classified each article based on the source it came from; that is, all articles from reliable sources are considered reliable, and all those from unreliable sources are considered unreliable.

We identified unreliable sources using OpenSources (www.opensources.co), which provides curated lists of categorized news sources. We included sources from the following categories: fake, satire, extreme bias, conspiracy theory, rumor, junk science, hate, clickbait.

As reliable sources, we used the following popular news sources:

```
npr.org, bbc.co.uk,
washingtonpost.com, nytimes.com,
abcnews.go.com, nbcnews.com,
apnews.com, reuters.com, usatoday.com,
cnn.com, foxnews.com,
huffingtonpost.com, wsj.com,
cbsnews.com, aljazeera.com,
theguardian.com, bloomberg.com,
politico.com, forbes.com, ft.com,
economist.com, theatlantic.com,
vox.com, pbs.org, csmonitor.com,
thefiscaltimes.com, latimes.com,
ocregister.com,
sandiegouniontribune.com,
nationalreview.com
```

We did not use strict guidelines when choosing these sources, but we attempted to include all of the most popular news sources that are regarded as reliable. We also ensured that sources with both conservative and liberal leanings are represented in our datasets.

For scraping, we used the webhose.io service, which downloads articles from specific sources. We used this raw data to construct two datasets: one containing the titles of the articles, and the other containing both the title and the body of the article, concatenated.

Each dataset (title and full) contained 4595 reliable articles and 4426 unreliable articles. These datasets were split into training sets, validation sets, and test sets, consisting of 70%, 20%, and 10% of the data, respectively. The reliable and unreliable articles were distributed in equal proportion across the sets.

As additional processing, we removed the name of the publishers from the titles of the articles. These were added by the scraping tool and resulted in our classifiers simply learning which publishers are reliable and which are unreliable. However, the bodies of the articles still contain references to the publishers, as well as other elements such as captions of images, social media markers, and even occasional HTML code. It was impossible to remove these elements efficiently. In addition, many of these would be seen by a reader of the actual article. Therefore, we left the bodies of the articles largely unchanged.

It is important to note that, in this situation, our classifiers do try to filter based on the publisher or other irrelevant elements. For example, some of the most predictive words found by our SVM are exactly publisher names such as ‘nbc’ and ‘fox’.

At a more fundamental level, the classifiers can only learn how to distinguish articles in our set of reliable sources from those in our set of unreliable sources. However, many of our results are broad enough to be useful in a more general setting.

3. SVM Classification

We first tried to classify using a linear support vector machine (SVM) model. Although we intended to use the model as a baseline for the other experiments, it turned out to have the highest accuracy. However, because of its rudimentary features, the model seems to have the lowest “generality”, as it focuses heavily on identifying the publisher of the article instead of finding insightful patterns in the text.

The SVM model uses bag of words features. Each article was transformed into a binary vector that had value one in all positions corresponding to words in the article and zero elsewhere. Words that appeared less than five times in the whole dataset were removed.

We implemented the SVM model using `sklearn` (the `sklearn.svm.SVC` model), with a linear kernel. The training and validation accuracies for various values of `C` (the inverse of the regularization coefficient), on both the title and the full data, are shown in Figure 3.1 and Figure 3.2, respectively.

C	Training accuracy	Validation accuracy
0.1	84.2	74.9
0.5	89.7	75.8
1.0	91.8	75.1
5.0	96.1	73.1
10.0	97.3	74.7

Figure 3.1: SVM accuracies for title dataset.

C	Training accuracy	Validation accuracy
0.001	92.6	89.6
0.005	97.2	92.4
0.01	98.5	92.7
0.05	99.9	92.5
0.1	99.9	92.1

Figure 3.2: SVM accuracies for full dataset.

The best validation accuracy for the title data was obtained with `C=0.5`. This gave 77.8% test set accuracy. For the full data, the best choice was `C=0.01`, which gave 92.6% test set accuracy.

Below, we show the ten most representative words in the best-performing SVM classifiers discussed above, for both reliable and unreliable articles. The words were selected as those whose position in the feature vectors had the highest and lowest weights after training, respectively.

Title:
Reliable: `two-way, virginia, can't, espanol, georgia, places, allegedly, miles, salt, series`
Unreliable: `pharma, breaking, update, attacks, !, approval, attempt, establishment, muslim, soros`

Full:
Reliable: `images, caption, read, photo, in, fox, it's, nbc, reuters, et`
Unreliable: `s, articles, !, 0, below, snip, credits, shares, sources, shop`

For the title dataset, many of the words are representative. In the case of reliable words, `allegedly` is a qualifier that we expect to be used much more in reliable articles than unreliable ones. The other reliable words, however, seem more specific to the news popular at the time when the articles were collected. In the case of unreliable words, many of them are related to conspiracy theories (`pharma, soros`), sensational language (`breaking, attacks`), or other topics regularly discussed in biased articles (`establishment, muslim`). It is also revealing that the exclamation mark is a word specific to unreliable articles.

For the full dataset, the words are less representative, and they often relate to the elements on the web page (`images, caption, read, photo, articles, below, credits, shares, sources, shop`). For reliable articles, we notice that publisher names are captured (`fox, nbc, reuters`). For unreliable articles, we notice again the exclamation mark.

4. Feed-Forward Network Classification

We implemented a fully-connected feed-forward neural network to gain more insightful classification behavior than the SVM. A neural network has the ability to learn feature transformations, as opposed to an SVM.

4.1. Architecture

The architecture of our feed-forward network is simple, because we wanted to achieve a baseline for neural model performance and test the importance of simply learning useful feature transformations without any specialized architecture decisions. Our network architecture is as follows.

1. The input is bag of words feature vector that is the size of the vocabulary, which is the same as the input to the SVM.
2. The input is fed into two fully-connected layers of sizes 320, and 64, with ReLU activations.
3. The output of the two fully-connected layers is fed into another fully connected layer of size 2, and then a log softmax activation function is used. The output models the probability of an article being reliable vs. unreliable.

We attempted to improve our performance by adding more layers, but the results only became worse, indicating that the problem of reliable vs. unreliable news classification is captured well by simple feature transformations.

4.2. Training

We used a variety of tuning techniques to achieve optimal performance. Our validation performance was the best when we did not use dropout layers or l2 regularization. The validation performance reached an optimum when we stopped training the net at about 15 epochs through the training dataset for the full data, and 49 for the title data. Unlike the SVM, the neural network could achieve perfect train accuracy, although the overfitting that occurs by the time the network learns this fitting degrades the validation accuracy.

4.3. Results

Figures 4.3.1 and 4.3.2 show our best results, given the architecture and training decisions from §4.1 and §4.2.

Train set score	93.2
Validation set score	76.7
Test set score	79.2

Figure 4.3.1: Tuned feed-forward net results on the title version of the data.

Train set score	99.2
Validation set score	92.7
Test set score	91.6

Figure 4.3.2: Tuned feed-forward net results on the full version of the data.

Below are the ten most representative words in each category, for models trained on both the title and the full data. We measured these by seeing which words, when used as input, lead to the highest confidence of classifications.

Title:
Reliable: `dodgers, sen, tax, young, salt, Virginia, ed, dies, way, shots`
Unreliable: `melania, eu, soros, Obama, November, antifa, pharma, video, muslim, breaking`

Full:
Reliable: `npr, unsupported, toggle, subscribe, autoplay, playback, et, nov, caption, speaks`
Unreliable: `anonymous, posted, media_camera, scroll, credits, below, articles, snip, shop, infowars`

These words may arguably appear to be more insightful than the SVM's top most influential words, but some still do not make much sense to a human judge; the words from the full dataset in particular are not incredibly insightful. For example, `autoplay`, `playback`, and `caption` do not give much information about the actual content of an article. Next, we examine recurrent neural network models to extract more meaning from news articles.

5. LSTM Classification

We implemented a Long Short-Term Memory (LSTM) network, in order to gain more reasonable classification behavior than our simpler models.

5.1. Architecture

Each word in an article was converted to its corresponding word embedding, according to a reduced subset of the GloVe word representations. Words without an embedding were discarded; this decision was made in order to keep the size of the input low. The sequence of word embeddings was then passed to the LSTM network. The network was bidirectional with dropout rate 25%. For the title dataset, we used hidden layer size and memory cell size 128; for the full dataset, we set the sizes to 200. The output of the LSTM at each step in the sequence was passed to a fully connected layer of size two, upon which a log softmax layer was applied.

5.2. Training

The training was slow and difficult, largely because of the massive input dimensions. The best models were trained on an Amazon Web Services instance with graphical units.

5.3. Results

Figures 5.3.1 and 5.3.2 show our best results, given our architecture and training decisions above. Although our LSTM classification accuracies are lower than the accuracies from our other classifiers, an examination of the LSTM model's weights reveals that it perhaps makes the most insightful decisions out of all of the classifiers.

Train set score	94.1
Validation set score	74.1
Test set score	73.8

Figure 5.3.1: Tuned LSTM results on the title version of the data.

Train set score	90.1
Validation set score	85.3
Test set score	84.1

Figure 5.3.2: Tuned LSTM results on the full version of the data.

Below, we show the model's most representative words. We measured these by seeing which words, when used as input, lead to the highest confidence of classification.

Title:
Reliable: county, raises, writers, arts, teachers, share, literary, literacy, directors, pen
Unreliable: liar, tearing, surfacing, uncut, jerking, twisting, revelations, dupe, screed, conspiracy

Full:
Reliable: outings, matches, portraiture, poetics, settings, scrappy, smokey, moody, spaces, expressiveness
Unreliable: scam, mongering, com, conspiracy, idiocy, liar, shameless, propaganda, exposing, ou

All the words seem very representative. For example, the reliable articles seem to have a lot of words related to culture, which we do not expect in unreliable articles (writers, arts, literary, literacy, pen, portraiture, poetics, expressiveness). The unreliable articles, on the other hand, consist mostly of sensational and provoking words, which we expect (liar, tearing, twisting, revelations, conspiracy, idiocy, shameless, propaganda, exposing).

6. LSTM Generation

In this section, we transition from classifying articles to providing a neural model that generates text based on unreliable and reliable news, to examine the respective stylistic differences more deeply than in previous sections. LSTMs can store arbitrary-length sequential information about bodies of text, so the LSTM model from §5 can be repurposed from a classification task to a generation task, with some architectural modifications. Because our computational resources are limited, our goal for this paper is to train a model to generate unreliable and reliable news sentences or statements, instead of full articles. The goal of the generated news statements is to give an interpretable standalone snippet of news, so such statements can be thought of as similar to headlines.

6.1. Architecture

Our LSTM model from §5 needs some changes in order to become generative. The new architecture is as follows.

1. The input is a 300-dimensional word embedding from the 1.9-million-word version of the GloVe embeddings. If there is no embedding for a word in the vocabulary, then a 300-dimensional sequence of zeros is passed in instead.
2. The input is fed to an LSTM, with a hidden layer about the size of the word embedding dimension (we used 360).
3. The output of the LSTM is fed into two fully connected layers with ReLU activations. The output of these fully connected layers is the size of the vocabulary and serves to decode the LSTM's representation of a word into an actual word from the vocabulary.
4. The output of the fully connected layers is passed through a log softmax activation, which provides a probability distribution over the words that the net can output. For generation purposes, the word with the highest associated probability is taken as the output.

6.2. Training

In order to train our LSTM model to generate coherent and meaningful news snippets, we had to design our training procedure with a few considerations.

For unreliable-style news, we selected 30 full articles from the unreliable news training set; we also followed a similar procedure when we trained the model to generate reliable-style news. We trained the model on such a small number of articles compared to our full corpus because training on all articles requires too much computing power. There are about 4000 unique words in 30 articles, but over 55000 unique words in all of the articles. Our generative model would have an output of size 55000 if we trained it on all of the articles. Furthermore, we do not need thousands of articles to get reasonable results, because our goal is to generate snippets instead of full articles.

We trained our LSTM by iterating through all sentences in the training articles (sentences were segmented from an article by splitting the article on the sentence-ending punctuation marks '.', '!', and '?'). We calculated the negative log likelihood loss based on whether, at each position in a sentence, the LSTM generated the correct word. The output of the LSTM depends only on previous words in the given sentence. This procedure allows our model to quickly capture the grammar and semantics of a sentence, which is perfect for our task of headline generation.

Finally, we found that it was useful to add a special 'stop' word after a sentence during training that signals to the network when it should stop the generation process. We could have made '.', '!', and '?' the 'stop' words, but sometimes humans will use several of these marks to add emphasis (e.g. '!?!!?') or to mean something else entirely (e.g. 'Dr. Minsky').

6.3. Results

When the network is trained per the specifications in §6.1 and §6.2, the results appear promising after a couple of hours on a CPU. Quantifying the quality of generated text is an important issue. One way is to use a perplexity measure or to devise another evaluation model, but such measures don't capture the semantic and grammatical intricacies of case inspections by a human, so we have opted to use a case inspection.

Generated text from training on unreliable and reliable news is shown in figures 6.3.1 and 6.3.2, respectively. The text is generated by inputting an initial seed word (the first word of the sentence) into the model, receiving the next word from the output, updating the hidden layer to include the new word, and continuing until the network generates the special 'stop' word. We searched the training corpuses for all of the sentences generated, to confirm that the model is indeed generating novel sentences; none of the generated sentences in figures 6.3.1 and 6.3.2 appear anywhere in the training data.

From an examination of the figures, it is clear that the structure of the generated language from unreliable news is substantially different than the case with reliable news. Our generative model indicates that unreliable news relies on offensive language, and grandiose claims with no specific sources (`scientists` instead of someone specific). In contrast, the generator that was trained on reliable news cites more specific sources (`he told reporters`) and does not use offensive language. Overall, our generative model yields the same insights that many of our classifiers found important in determining whether an article is reliable or unreliable.

Aliens need to go much deeper on Trump and Cheney.
Liberal, will be those who cannot or will not resist.
China is dead.
Liberals are ugly Jewish females with hooked noses.
The Jews have used chemical weapons.
Obama won't have a retirement plan.
Alcohol of antioxidants to become a key part of the chemical as part of the hearings.

Scientists are caught between drowning and freedom.
Trump was arrested by Casper police after he spent 804 days in space thanks to time dilation.
Chinese dream is now killing the Spanish police in the Spanish Empire.

Figure 6.3.1 Sentences that our LSTM model generated after it was trained on unreliable news articles.

Trump and Xi met with on his bus at a bus after detecting the smell of marijuana, but he was arrested or put in jail.
American president President Trump vowed to deport millions of people in the country illegally.
A person familiar with refugee camps, Ai Weiwei ponders the question.
Clinton's resignation was intended to break our spirit, he told reporters.
Xi promised to comment.
After the crash, the father says they are making it bigger.
One killed at Colorado 2015.
Globalization has been charged research led by the scene.
The star took to twitter where he explained his side of the story in jail.
That person was not to publicly discuss the internal meeting.

Figure 6.3.2 Sentences that our LSTM model generated after it was trained on reliable news articles.

7. Conclusion

As fabricated online news stories become more prevalent, the importance of an automated way to detect such stories becomes greater. We have explored a variety of machine learning approaches to classify, generate, and extract insightful information from news articles, with the goal of determining which news articles are reliable or unreliable. We have found that there are stylistic patterns that reveal a great deal about whether articles come from an unreliable source, such as offensive language and unsubstantiated claims. Such differences allow our classifiers to achieve as much as 85%-92% accuracy on classifying full articles and around 75% accuracy on classifying articles by title alone. These differences also allow our generative model to learn and generate very different language styles depending on whether it was trained on reliable or unreliable news.

Overall, our work has highlighted the structure of unreliable news, and has provided techniques by which it can be identified.

Our code is available publicly on GitHub at github.mit.edu/rbuhai/6867_final_project. We used the `sklearn` and `pytorch` libraries for the implementation of all models.

8. Division of Labor

Rares Buhai worked on data collection, the SVM classifier, and the LSTM classifier. Tristan Thrush worked on the Feed-Forward Neural Network classifier and the LSTM generator. The paper was written collaboratively, although Rares and Tristan each focused on writing sections that correspond to their respective contributions.

References

[1] opensources.co
[2] pytorch.org
[3] scikit-learn.org
[4] webhose.io