Detection of Satirical Headlines

Evan Fellman and Ishita Kumar

Abstract

Our goal is to make a computer program that is good at detecting when an article headline is satirical or not. We tested several different models to see how each performed. Our goal was an accuracy of 80%. This is a difficult task for humans so we expected this to also be difficult for computers. We were pleasantly surprised that we were able to get an accuracy of 86% by using a long short-term memory model with a bag of words model for features.

Introduction

Given current events, teaching a computer how to tell the difference between a satirical article and a serious one is an incredibly relevant task to many companies such as Facebook. Facebook and many other companies are now implementing strategies to label fake news articles and satirical articles to stop the spread of misinformation [1]. Our goal with this project is to have a computer be able to tell the difference between a satirical article headline and a serious headline.

Detecting satire has always been a difficult problem for many. Even humans sometimes are not the most accurate at detecting whether an article is satirical. Our aim is to find the best model and type of features to analyze to be able to differentiate between satirical article headlines and non-satirical article headlines. This is a difficult problem to solve. For example, we can find the following headline in the dataset that we use for this experiment:

North Korea 'Feeding Workers Crystal Meth' to Speed Up Skyscraper Project

Most people would determine that this headline must be satirical. We find that a lot of satirical headlines have outrageous words and things that just don't seem realistic. Generally speaking, the phrases "North Korea" and "crystal meth" generally are followed by something outrageous or are inherently outrageous. Also, the idea of a country drugging its citizens to complete a construction project feels ridiculous and not real. Yet, this is not satirical. We were shocked to hear it wasn't. The above headline is very real and it truly happened. This was the headline of an article from the *Huffington Post*. Considering that even humans, despite our much better understanding of language and the world around us, can classify these headlines incorrectly, to get a computer to correctly determine that this headline is not satirical would be impressive.

Related Work

We found two papers that are relevant to the task at hand. The first, by Andreas Stöckl, completed a similar task as they aimed to detect satire in the news using machine learning models. Specifically, they used logistic regression and linear support vector machines. They were able to achieve “a precision of 98.7% and a recall of 95.2%” [2] which is seriously impressive as many humans struggle with detecting whether an article is satire.

Something to note is that their task was slightly different than ours. They analyzed the entire article, while we are just analyzing the headline. Part of the reason why we are only analyzing the headline is to save processing time and it requires less computational resources to be able to handle that large amounts of data. We were unable to get as great results from our experiment. This could be because of how we only analyze a headline as opposed to their analysis of an entire article.

The second paper also accomplished a task that is similar to our goal. They wanted to detect if a tweet was satire. This paper only analyzes tweets in Spanish, however. This is still relevant as we do not expect the difference between English and Spanish to impact the results and effectiveness of the models significantly. This team was able identify if a tweet was satirical based on the usage of “frequency, ambiguity, part of speech, synonyms, sentiments, characters, and slang words” [3]. They managed to achieve an accuracy of 81.4%. Although not as impressive as the first paper, this is still a difficult accuracy to achieve.

In a way, this is more similar to our experiment because it analyzes tweets. This is similar to headlines because both are short summaries or assertions frequently. This does differ quite a bit from our experiment as well because tweets tend to follow a different format and have a different language than headlines do. Also, the motivation between a tweet and a headline is different. Frequently, a tweet’s goal is to be read while a headline’s goal is to get people to click on the article and read the article that the headline corresponds to.

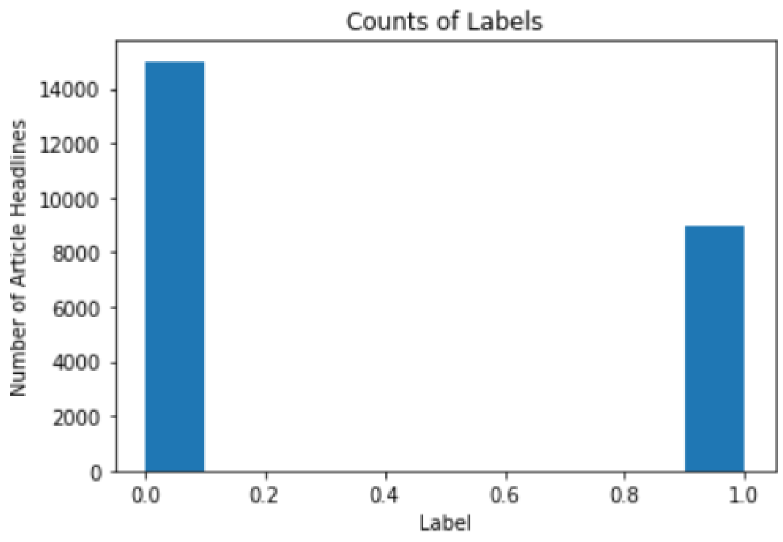
Data

To accomplish this goal we have a data set of headlines from *The Onion* to act as the headlines for the satirical articles and we have a collection of article headlines from the website reddit.com/r/nottheonion to act as the headlines of serious (non-satirical) articles.

Luckily to save time in annotating data, we are able to pull datasets from online that come pre-annotated. A popular choice for these datasets is kaggle.com. We found one dataset that has two collections: one that is a collection of article headlines from reddit.com/r/nottheonion and the other is a collection of article headlines from *The Onion*. We found our dataset at https://www.kaggle.com/chrisfilo/onion-or-not. *The*

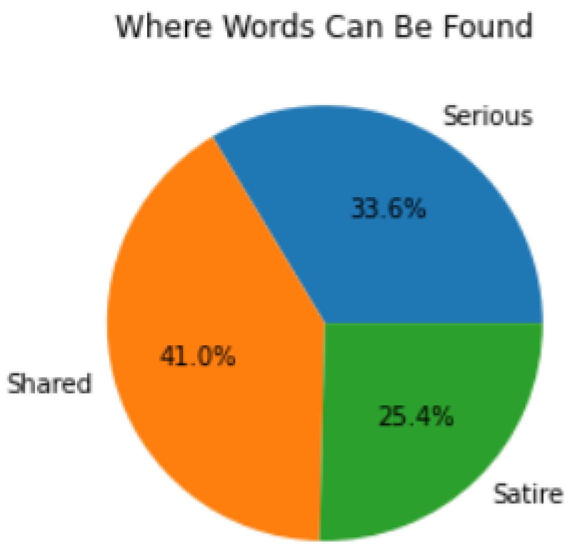
Onion is a famous satirical website. Reddit.com/r/nottheonion collects real (non-satirical) headlines that might sound as if it came from *The Onion* but are in fact true. Later we give an example of one of these headlines. The dataset we are using got all of its data using github.com/lukefeilberg/onion.

It saves us a considerable amount of time for many reasons. The main way it helps save time is through its labeling: it labels each headline with a 0 if it is a serious headline and labels a headline with a 1 if it is satirical. None of the data is mislabeled nor unlabeled. Therefore, we have no reason to discuss how to interpret any data that is unlabeled. In total we have 24,000 headlines; 9,000 headlines are from *The Onion* while 15,000 headlines are serious headlines.



In the above graph, we can see the number of serious (a label of zero) and satirical (a label of one) headlines. We can also see that there are only these two labels as these are the only two subjects that have any headlines in them. If there were other labels then we would see other bars in this graph. But we do not have any other bars (and we ran a script) so therefore we know that these are the only labels.

We choose this dataset (as opposed to others) for a variety of reasons. The size of the dataset is large enough that it allows a good split between testing and training examples. Therefore we have enough data to train on to avoid underfitting. The data present was also clean as well as properly labeled. This allows us to have a reduced amount of noise when training our model. This also lowers the amount of work that we have to do. We can simply trust that each row was labeled correctly, as opposed to having to determine where the headline was from in order to label it.



Above we can see by proportion how many words are unique to satire headlines and unique to shared headlines. Not surprisingly, we have a 41% overlap of words used in satire and non-satire article headlines. This is not surprising because many topics are touched on in both satirical and serious article headlines. For example, the word “president” is likely shared between the two because both discuss things related to the president.

Luckily, there was no need that we discovered to make any changes to the dataset. However, if we had discovered a need to make a change, whether that is by editing an entry or adding new ones, then, annotating the values should be a fairly straightforward task. This is because each row only contains two pieces of data: the headline and the label. To edit a row or make a new one, we would simply have to correct or make the headline or put in the correct label (either zero if it is serious or one if it is from *The Onion*). To determine if it is from *The Onion* we can simply search their website for that article.

Method

In many natural language processing experiments, there is a data collection and annotation phase. Luckily for this experiment, neither was necessary thanks to the database that was discussed before.

We wrote our experiment in Python because we found the most documentation for the models that we wanted to use in Python. We needed to split our data into training and testing. So we used sklearn library’s train_test_split function. We train on 60% of the data and then we test on 40% of it. No matter the features or models we use, we will keep this constant. We chose this because we believe that this gives the best accuracy while running models while also having enough data to test on. Since the dataset comes in a seemingly random order, all of the *Onion* headlines are mixed in with the serious headlines, there is no need to shuffle while splitting the dataset into the two parts. We want to randomly divide up the data but we also want to make sure that it is repeatable. So we use the default random system but we provide a constant random state. We use the same random state throughout our project: 42.

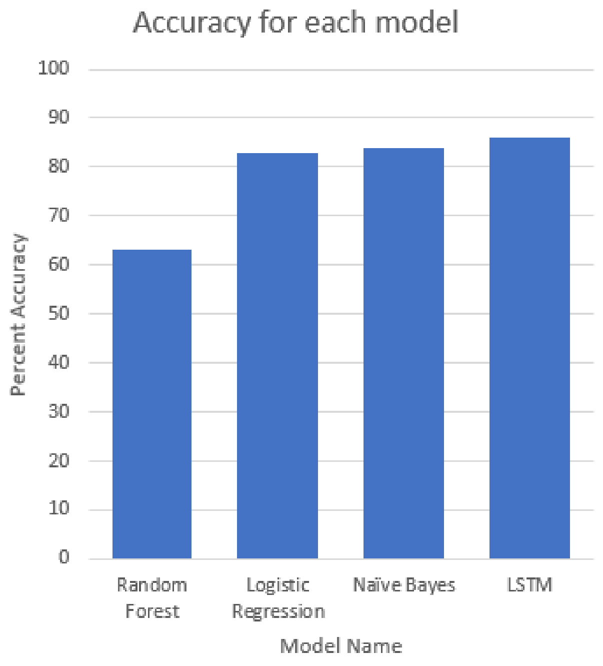
Before splitting up the data, we removed the stop words to help remove the noise. We did not perform lemmatization and this would likely improve accuracy. So if this experiment were to be repeated, this would be a good addition.

From here we would train then test each model: a random forest model, a logistic regression model, a naive bayes model, and a long short term memory model. We get each model prebuilt (not pretrained) from the sklearn library. We then analyze their accuracies. With the naive bayes model, we decided to also analyze which words are most indicative of a headline being satirical due to the simplicity of that analysis and the high accuracy of the naive bayes model.

Results

To consider this experiment a success, we want to see at least one combination of a model and system of features that has a 80% accuracy in classifying article headlines. This 80% goal was determined through an arbitrary means as we considered 80% to be a high number.

We have implemented various natural language processing models that result in different accuracy values. So far we have implemented a random forest model, a logistic regression model, a naive bayes model, and a long short-term memory model. Below are the accuracies for each model:



The random forest model had an n_estimator of 100. That means that there were 100 trees in our forest; while training our model, we decided the best prediction for each case was the maximum value given by all of the trees. We chose to use information gain instead of gini to decide how to choose which feature we wanted to split on. We did this because information gain performed better than gini. The maximum depth of each tree was 2 nodes deep.

The accuracy of the random forest model was 63%. This is not what we were hoping for. But this was not surprising either. The size of the dataset likely caused us to have a much lower accuracy than desired. Still, this is significantly better than guessing. So when we saw this accuracy we were satisfied with this result. Random forest works better with bigger datasets. Our dataset only contains 24,000 rows and only 60% of that is used to train on. This is not enough for random forests to be good at classifying. Therefore we were not surprised to see that this model did not work well.

We performed much better with our second model, logistic regression. This model achieved an 83% accuracy. Here we did not have any significant values or variables here that are not considered to be the default. This is significantly better than the random forest model. This is an excellent accuracy. When we saw how well it performed, we felt confident that we can do even better as we expect the long short term memory model to perform much better.

In the theme of performing better with every attempt, we used a naive bayes model third. Again, we did not have any significant values or variables here that are not considered to be the default. This model only marginally beat out our logistic regression model by 1%. Naive bayes got an accuracy of 84%. We did not expect naive bayes to beat out logistic regression.

Rank	Word	P(Satire|word=Word)
1.	ftw	0.9638421967692736
2.	heartwarming	0.9615244227107462
3.	blog	0.9608977939423082
4.	announced	0.9574313423096154
5.	fucking	0.956922156163339
6.	awesome	0.9503953329877892
7.	heartbreaking	0.9420082550213372
8.	quiz	0.9420082550213372
9.	cons	0.940572848159645
10.	pros	0.940572848159645
11.	patriothole	0.9374777574353088
12.	clickhole	0.9374777574353088
13.	dnc	0.9340425553640017
14.	shit	0.9321793986646605
15.	incredible	0.9302079217702436
16.	struggling	0.9281183955872977
17.	asshole	0.9258998899334995
18.	exhausted	0.9210250287901169
19.	clearly	0.9210250287901169
20.	hasn	0.9210250287901169

Although our naive bayes was not the most accurate model we have, it is significantly easier to analyze than our most accurate model and it has a very close accuracy to our most accurate model's accuracy (a difference of only 2% accuracy). It is significantly easier to analyze than a long short term memory model because we can analyze the probability of a word being used in a satirical headline as opposed to trying to analyze a neural network. Analyzing neural networks is an incredibly difficult task. Since you would have to analyze all of the behaviors of each of the neurons in each of the layers. This task would deserve a paper to itself. Therefore since this complexity of the analysis would stray away from the goal at hand, we analyze the words' behavior in naive bayes. Above we can see the rank of each word along with the probability of it being satirical.

These results are not surprising for several reasons. The top world "ftw" is not surprising because generally, legitimate news articles will avoid using informal phrases such as "for the win." The numerous curse words were also not surprising as I would never expect to see a curse word in a legitimate news article headline. In general informal words are not surprising to see be more used in satire. What was surprising was to see words like "dnc." I would expect this word to be considered to be a lot more serious. As I would expect that many legitimate news headlines would discuss the democrats and refer to them as the "dnc." Another word that was surprising to have such a high chance of being satirical is the word "feminism." "Feminism" had a probability of 90.9% of being satirical.

We were able to beat our accuracy from naive bayes with our long short term memory model. This was able to achieve the high score of 86%. For our long short term memory model, we used the Keras library instead of sklearn. This is not for many reasons; it was just simply the one that we felt was most intuitive to use. We use sigmoid as our activation function. Other than this we use default values.

Discussion and Future Work

Overall, we feel impressed with the accuracies that we got. Humans are much better at processing natural language than computers are with the current state of computing. We humans have had a much longer time to evolve to improve at this. Therefore we were fully ready and content to write the paper on the difficulties of detecting satire and why it was simply just not possible to hit our goal of 80% accuracy. Luckily, this did not have to happen. We managed to pass our goal with three separate models.

We believe that, although our accuracies are very high and very exciting, this is not high enough to employ on something as large and as important as Facebook or Instagram. There are problems that we need to improve. For one, we only tested on *The Onion* articles. We would argue that this implies that there is a large chance that it would be less accurate at detecting satirical article headlines that are not from *The Onion*.

We also did not run this experiment to see if this can detect nonsatirical false news. Sadly, frequently false news is spread quickly. This news is not always written with the intention of satire. *The Onion* purposely writes ridiculous articles (and headlines as well) that are outrageous or funny. But nonsatirical false news does not have this intent and this can change how the headlines are written. A fantastic example of this is the PizzaGate story [4]. This is a completely false and dangerous story that is clearly not written with satirical intent [4]. Therefore since the intent behind writing the articles and headlines are different, our trained models may fail to detect this as fake. It would, correctly and simply, determine that this is not satire, which it isn't.

Another reason this needs further development before implementing it in a large social media such as Facebook is because it isn't accurate enough. Although an accuracy of 86% is incredible, there is still room to grow. It is entirely possible that an important and real article gets marked as satire by mistake by this algorithm. This could then lead to some denying true facts which is dangerous and is the very thing that we are trying to prevent. Personally, we would like to see an algorithm get at least 95% accurate before it is launched on such a large and potentially dangerous scale.

Despite these potential issues of the experiment, this has shown great results that show that computers can truly be good at determining whether an article is satirical purely based on the limited information that a headline gives you.

There are a lot of ways to improve this. Specifically, in that, we used mostly default values for our models. We strongly believe that if we tinkered more with the settings of these models, we would be able to achieve even more impressive results. Also if we had experimented with word embeddings then we may have had better results as well.

References:

[1] Veronica Penney. 2020. *How Facebook Handles Climate Disinformation*. (September 2020). Retrieved October 1st, 2020 from https://www.nytimes.com/2020/07/14/climate/climate-facebook-fact-checking.html

[2] Andreas Stöckl. 2018. *Detecting Satire in the News with Machine Learning*. University of Applied Sciences Upper Austria, Hagenberg, Austria.

[3] Francesco Barbieri, Horacio Saggion,Francesco Ronzano. 2015. *Is This Tweet Satirical? A Computational Approach for Satire Detection in Spanish*. Universitat Pompeu Fabra, Barcelona, Spain

[4] Cecilia Kang. 2016. *Fake News Onslaught Targets Pizzeria as Nest of Child-Trafficking*. (November 2016). Retrieved December 1st, 2020 from https://www.nytimes.com/2016/11/21/technology/fact-check-this-pizzeria-is-not-a-child-trafficking-site.html