Classifying Real and Satirical News: r/theOnion, r/nottheonion, and r/news
Final Report

Melanie Malinas
Springboard Data Science Career Track
Capstone Project 2

Introduction and Problem Statement:

The issue of real news vs. fake news has come under increasing scrutiny in American society over the last few years. It is a huge problem that many data scientists are currently working on. I wanted to address a possibly related problem: detecting satirical news. The Onion, a satirical news website, has become very popular in the past several years, and I have often seen people on social media who sometimes don't realize that the Onion is satirical and believe that it is true. On the website Reddit, there is a message board (subreddit) that is devoted solely to posting real news articles that people think sound like Onion articles. For this project I decided to create a machine learning model that could distinguish between posts on this subreddit, r/nottheonion, from posts on the subreddit r/TheOnion, which posts Onion articles. I also wanted to see how well you can distinguish between the Onion and regular news from r/news.

Data Cleaning and Processing:

I obtained my data using PRAW, the Python Reddit API Wrapper. PRAW made it very easy to access the Reddit data. I was able to access the top 1000 (approximately) posts from r/nottheonion, r/TheOnion, and r/news and save them as CSVs. I decided to do my analysis solely on the headlines and not on the actual text of the articles, because oftentimes people only look at the headlines.

I then used various methods of natural language processing (NLP) to process the data and get it ready for the machine learning model. I primarily used the NLP libraries NLTK and spaCy to process the data. With spaCy, I did something called named entity recognition, which is when you classify "named entities", i.e. people, places, organizations, etc. spaCy's libraries are statistical neural network models. Using spaCy, I identified the named entities and their named entity categories and removed them from each headline.

In order to process the headlines without the named entities, I took several steps. I first made the headlines all lowercase, removed punctuation (using regular expressions), and removed numbers (also using regular expressions). I then tokenized each headline using NLTK, splitting up the headline into word tokens. Then, for every word in the list of word tokens, I used NLTK to lemmatize the word. Lemmatization aims to reduce a word to its base form by getting rid of inflection. I chose lemmatization over stemming because stemming often produces tokens that are not actually words, whereas lemmatization always returns valid words. I then used NLTK to remove stopwords, which are very common words such as "it", "and", "the", etc. which do not really give information.

I then took the named entities that I had identified using spaCy and added them back to each title along with their named entity types, so that I could use all of the information to classify the headlines.

Machine Learning and In-Depth Analysis:

I then created a machine learning model that would learn to distinguish between Onion headlines and either r/news headlines or r/nottheonion headlines. I decided to use a Naive Bayes model, which is commonly used in the NLP field and is simple and easy to work with. I implemented Naive Bayes in scikit-learn. I optimized for the best alpha hyperparameter using a custom CV score function which I defined and attempting to optimize the F1 score. Using an alpha of 1, I achieved an accuracy on the training data of **0.85** and an accuracy on the testing data of **0.75**, with an F1 score of **0.72**, for Onion vs. nottheonion headlines. I then decided to test whether there would be an improvement if I used bigrams instead of just single tokens. I found that this was a slight improvement: though the accuracy on the training and test data was the same, the F1 score for bigrams was **0.73**.

I then looked at the top words for Onion vs. nottheonion headlines as well as the most strongly misclassified headlines. Some of the top words/bigrams for the Onion were “trump”, “regularly”, “regularly happens”, “say nation”, and “announces”. Some of the top words/bigrams for nottheonion were LOC (the location tag for named entity recognition), “officer”, “protest”, “China”, “Texas”, and “police”.

These were some of the most strongly misclassified sentences:

```
Actually not the Onion but mis-classified as the Onion
---------------------------
‘Live pee or die’: N.H. governor steps in to let woman keep her ‘PB4WEGO’ license plates

White supremacists taking DNA tests sad to discover they’re not 100% white

Losers are more likely to believe in conspiracy theories, study finds

DeVos backlash Sees Parents Threatening to Homeschool Kids

Tobacco smokers could gain 86 million years of life if they switch to vaping, study finds

Actually the Onion but mis-classified as not the Onion
--------------------------
Dallas Cops Plant Black Suspect At Murder Scene

‘C’mon, C’mon,’ Says Matt Damon Desperately Searching For Own Name On List Of IMDB User Dolphinsoul60’s Top 100 Actor
s

Blog: If You’re Not A Police Officer, You Can’t Understand The Pressure You Feel In The Split Second When You Have To
Decide Whether Or Not To Shoot An Unarmed Civilian 8 Times

Heartbroken Russian Ambassador Thought Special Meetings With Jeff Sessions Were Very Memorable

Trump Confident U.S. Military Strike On Syria Wiped Out Russian Scandal
```

As you can see, some of the headlines in nottheonion that were misclassified really seem like they could be satirical news headlines. Headlines that were misclassified that are actually the

Onion do not seem like they could be real news, but it is clear that words like “police” and “cops” cause the model to misclassify the headline.

I then trained another Naive Bayes classifier to distinguish between Onion headlines and headlines from r/news, i.e. headlines that do not necessarily sound like satirical news. Doing this with bigrams in a similar manner to the r/nottheonion headlines, I found that this classifier was even better at distinguishing between Onion headlines and real news headlines than the classifier that distinguished between Onion headlines and nottheonion headlines: with an alpha of 0.1, the accuracy on the training data was **0.88** and the accuracy on the test data was **0.81**, with an F1 score of **0.79**. This means that it is even easier to distinguish between Onion headlines and regular news headlines, which is perhaps unsurprising. Some of the top words/bigrams for r/news headlines were “sue”, “fired”, “California”, “judge”, and the named entity type of ORDINAL.

These were some of the most strongly misclassified sentences:

```
Actually news but mis-classified as the Onion
---------------------------
The Italian government has approved a law ordering parents to vaccinate children or face fines. The authorities have
noted a rise in measles cases, which the cabinet blames on "the spread of anti-scientific theories."

Decade in the Red: Trump Tax Figures Show Over $1 Billion in Business Losses

George Clooney Calls for Online Release of 'The Interview "That's the most important part. We cannot be told we can't
see something by Kim Jong Un, of all f***ing people."

Facebook "allowed Microsoft's Bing search engine to see the names of virtually all Facebook users' friends without co
nsent, the records show, and gave Netflix and Spotify the ability to read Facebook users' private messages."

Millennials earn 20% less than Boomers did at same stage of life

Actually the Onion but mis-classified as news
--------------------------
Man Who Crossed US In Balloon Only Talks About Horse Abuse

PR Firm Advises U.S. To Cut Ties With Alabama

Roadmap To Peace: Necco Has Set Aside A Roll Of Wafers For Israel And Palestine To Share Only After They Achieve A Tw
o-State Solution

‘You Are Donald Trump, 45th President Of The United States,’ Trump Reads From Faded Tattoo On Wrist

Trump Confident U.S. Military Strike On Syria Wiped Out Russian Scandal
```

Developing a Web-based App:

I then developed my r/theOnion vs. r/nottheonion model with bigrams into a web-based application using Flask and Google Cloud App Engine. I converted my Jupyter notebooks into .py files to be used on the website, which involved making almost all of my code into functions. In order to make my website fast, I trained the model on my own machine, then pickled the trained model to be used on the website and imported it into another file which was then used in the Flask code. The way the website works is that you paste in an article headline from either r/TheOnion or r/nottheonion and then it gives you probabilities that that headline is either the Onion or nottheonion. You can also try it with other words or phrases which can provide information on how individual words or phrases contribute to the model - for example, the word “nation”, which is often used in Onion headlines, comes up as having a 0.93 probability of being the Onion and only a 0.07 probability of being nottheonion. The word “dog” has a 0.7 probability of being nottheonion and and 0.3 probability of being the Onion. You can check out the website at https://onion-nottheonion-app.appspot.com/.

Conclusion and Next Steps:

This project demonstrated that it is possible to create a model that can distinguish between Onion articles and real news articles, simply based on Naive Bayes with word tokenization and named entity recognition. However, I think that it is important to recognize that the Onion often writes their headlines in a distinctive way, which makes it easier for the model to pick up on it. I therefore think it would be interesting in the future to look at other satirical websites besides the Onion as well to see how well that worked. Another important point is that the top 1000 Onion articles and the top 1000 nottheonion articles cover very different topics. It is clear, for example, that when an Onion article mentions the police it is much more likely to be classified as a nottheonion article, because there are many more nottheonion articles about the police. Building a model that could distinguish between satirical and real news regardless of topic would perhaps be a more challenging project.