Group 6 Discussant Report

CS 216

Github link: https://github.com/jasoncode/cs216project

Summary

Especially leading up to and since the 2016 election, there has been a lot of discussion about “fake news.” It has come to the point where it is a politically charged term that anyone can use it to discredit a news outlet that disagrees with them, and social media sites are launching efforts to combat its spread. Our project will analyze the language used between real and fake news sources, especially in regard to sentiment analysis.

Because it was the 2016 election that brought fake news into the spotlight, we will be analyzing articles on the topic of politics to get the most relevant and interesting results. We expect to find differences in the language used between real and fake news, specifically that fake news will contain more emotionally charged language than more reputable news sources.

Data Collection

Complications so far have been in defining and identifying fake news. Some sites intentionally mix real and fake news, which affects our ability to delimit the sites. Also, many sites include a disclaimer that nothing they write should be trusted. If a site gives up its hoax, is it still fake news,or satire? These questions have led us to reconsider how we will visualize our data, because it may not be wise to clump all fake sites into one category. Therefore, we are attempting to analyze only those news sites that claim legitimacy, though this is proving to be a challenge.

In addition to the fake news sites we scraped, we also used the fake news dataset provided by Kaggle. This was more difficult to use than we anticipated, as the dataset was full of satire, rather than fake news presented as real news, non-political news, and news in other languages. We were able to hand-clean this dataset, leaving us with 185 new sources of fake news. These sources ranged from only 1 or 2 to as many as 489 articles scraped.

‘Fake’ Sources
Your News Wire
Counter Current News
Kaggle Sources

Real News Sites
Wall Street Journal
Fox News
NYT

The Guardian
Breitbart
Economist
CNN
Washington Post
Drudge Report
The Blaze

Scraping

We are using python to acquire data from news sources. The first step to acquiring news articles is getting a list of URLs to articles we want to use. For some sources this will be possible through an API call, such as the New York Times. For most we will need to scrape these URLs from the website itself. Once we have URLs to a sufficient number of articles (currently we are getting 500 from each website), we will scrape the article body text from the website. We are using the python library lxml, in combination with xpath, in order to scrape data. These tools allow us to easily access specific sections of the html and acquire all text from that section and child sections. For example, many articles are constructed with sequential <p> tags, often with child tags for things like links. Lxml and xpath allow us to acquire all of the text in just a few lines. We then encode the text to normalize formatting and write it to a text file. This text file is then used as input for the sentiment analysis.

While scraping articles is in an effective and relatively simple method for getting articles in bulk from a specific site, there are some challenges for a project like this where we need variety. These mainly revolve around the fact that every website is a little bit different, so it is difficult to code a scraper that works for every website. In order to make this simpler, we wrote a util that manages requests and error checking, as well as some of the logic behind scraping articles in bulk and formatting them to text files. However, since every site has different HTML and text formatting, every site needs to be examined manually using browser dev tools in order to manually write the xpath. Additionally, we occasionally need to do some extra text processing for sites that have sections that shouldn't appear in the article text.

Methodology: Sentiment Analysis

We have downloaded a Python library called textblob which enables sentiment analysis based on its extensive corpora. The documentation for the library is at the following link for additional information on the library as a whole: https://textblob.readthedocs.io/en/dev/. The key aspect for our project is the sentiment feature, which returns a two-element tuple consisting of polarity, on a scale of -1 for most negative to 1 for most positive, and subjectivity, on a scale of 0 for most objective to 1 for most subjective. The file textblob_test.py includes several intuitive example sentences demonstrating the functionality.

The analysis file computes the sentiment for each article collected (as described above). It outputs three files which report the data in different ways:

- Results_ByArticle.csv - This file contains a row for each article containing the source name, polarity, and subjectivity, as well as summary statistics for each overall category of news
- Results_BySource.csv - This file lists results article by article but listed under a clear header for its source, as well as summary statistics for each source and each overall category of news.
- Results_Source_Summary.csv - This file contains a row for each source, listing the number of articles from that source and summary statistics. Because this only looks at summary statistics, sources are only included if they contain at least five articles.

As noted above, much of the data output hinges on summary statistics. The statistics we chose to output are as follows:

- Mean
- Absolute Value Mean - This one was included because polarity is on a scale from -1 to 1. Thus, if a source had a lot of articles that were polar but on opposite ends, the mean could come out close to 0, which would not accurately reflect the polarity. In this case, if most of a source's articles
- Median
- Absolute Value Median - Included for the same reason as absolute value mean

Analysis and Visualization

Our primary form of visualization is scatter plots. These plots place subjectivity on the x-axis and polarity on the y-axis, creating a clear picture of the sentiment analysis of the articles we have gathered. Due to the nature of the output, there will be several different scatterplots we can make. The most important one will be a scatterplot of the sentiment analysis of all of the articles that we have scraped. This should give us a big picture view of the nature of the data we are dealing with. Once complete, these will hopefully reveal trends about the overall nature of the data which we can use to guide our conclusions. If they do not reveal trends, this will be an interesting result as well, however. It will counteract some of our most fundamental assumptions about fake news and how easy it is to identify, and raise questions about fake news detection going forward.

We will also be able to make visualizations of the summary statistics over each source. While we will not have enough sources to make any statistically significant conclusions, we will be able to see trends and make commentary on the data that we do have.

We are also going to make a word cloud of the most common words seen in political articles in real and fake news. We would expect many of the major words to be the same (Trump, Obama, Hillary, Putin, etc.) but it is also possible that some clear differences in second or third-tier words will reveal additional information about the nature of fake news in today's political climate.

Group 6 Presentation Outline

Intro [names]:
- Problem Description & Hypotheses

Datasets: [3]
- Scraping/cleaning/conversion
- Challenges & Adjustments

Classification/analysis: [2]
- Explain libraries used and how
- Explain information outputted and data

Visualizations: [2+]
- Fake/Real News comparisons (scatter plots)
- Word Cloud

Conclusions/Takeaways

Q&A