Fake news detection

Reported by Group-11,

29 April, 2023

Now-a-days we encounter with many fake news posts in Facebook and believing them without knowing that are fake. Nonetheless, these fake news posts have exploited Facebook users' feeds to propagate throughout the internet. So the data science community has responded by taking action to fight the problem. There's a Kaggle-style competition called the "Fake News Challenge" and Facebook is employing AI to filter fake news stories out of users' feeds. Combating fake news is a classic text classification project with a straight-forward proposition: *Can you build a model that can differentiate between "Real" news vs "Fake" news.* And that's exactly what we attempted to do for this project. we assembled a dataset of fake and real news and employed a Naive Bayes classifier in order to create a model to classify an article as fake or real based on its words and phrases.

When we first started this project, we conceded that this would not be the perfect project. The purpose of this project was to see how far we could get in creating a fake news classification and what insights could be drawn from that, then used towards a better model. Nonetheless, we expect some valuable insights to come from this project.

Since this is a text classification project, we only used a Naive Bayes classifier as is standard for text-based data science projects. The real work in formulating a model was the text transformation (count vectorizer vs tfidf vectorizer) and choosing which type of text to use (headlines vs full text). This gave us four pairs of reconfigured datasets to work with. The next step was to determine the most optimal parameters for either a count-vectorizer or tfidf-vectorizer. This means using a n-number of the most common words, using words and/or phrases, lower casing or not, removing stop words (common words such as the, when, and there) and only using words that appear at least a given number of times in a text corpus (a term for a text dataset or a collection of texts). To test the performance of multiple parameters and their numerous combinations, we utilized the Sci-kit Learn's GridSearch functionality to efficiently execute this task. After the grid search cross validation process, we found that my model worked best with a count vectorizer instead of a tfidf and produced higher scores when trained on the full text of articles instead of their headlines. The optimal parameters for count vectorizer are no lowercasing, two-word phrases not single words, and to only use words that appear at least three times in the corpus.

There were two parts to the data acquisition process, getting the "fake news" and getting the real news. The first part was quick, Kaggle released a fake news dataset comprising of 13,000 articles published during the 2016 election cycle. The second part was… a lot more difficult. To acquire the real news side of the dataset, we turned to All Sides, a website dedicated to hosting news and opinion articles from across the political spectrum. Articles on the website are categorized

by topic (environment, economy, abortion, etc…) and by political leaning (left, center, and right). I used All Sides because it was the best way to web scrape thousands of articles from numerous media outlets of differing biases. Plus, it allowed to me download the full text of an article, something you cannot do with the New York Times and NPR APIs. After a long and arduous process I ended up scraping a total of **5279 articles**. The articles in my real news dataset came from media organizations such as the New York Times, WSJ, Bloomberg, NPR, and the Guardian and were published in 2015 or 2016. We decided to construct our full dataset with equal parts fake and real articles, thus making my model's null accuracy 50%. I randomly selected 5279 articles from my fake news dataset to use in my complete dataset and left the remaining articles to be used as a testing set when my model was complete. Our finalized dataset was comprised of 10558 total articles with their headlines and full body text and their labels (real vs fake).

The true test of the model's quality would be to see how fake news articles in the test set (those not used in the creation of my model) it could accurately classify. Out of the 5234 articles left in the other fake news datasets, our model was able to correctly identify 88.2% of them as fake. Building a model based on a count vectorizer (using word tallies) or a tfidf matrix (word tallies relative to how often they're used in other articles in your dataset) can only get you so far. These methods do not consider important qualities like the word ordering and context. It's very possible for two articles that are similar in their word counts to be totally different in their meaning. We did not expect our model to be adept at handling fake and real articles whose words and phrases overlap.

In conclusion, while we think that a standard Naive Bayes text classification model can provide insight into addressing this issue, a more powerful deep-learning tool should be employed to combat fake news in a professional setting. Classifying "fake news" provides a novel challenge to the data science community. In many machine learning projects, the distinction between the different classes you want to predict is clear, whereas it's a lot murkier in this case. This project validates the notion in data science that intuition and intimacy with your data is just as or more important than any model or tool at your disposal.

References:

http://www.fakenewschallenge.org/

https://qz.com/843110/can-artificial-intelligence-solve-facebooks-fake-news-problem

https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html

https://www.kaggle.com/datasets/mrisdal/fake-news

https://www.allsides.com/unbiased-balanced-news

https://machinelearningmastery.com/how-to-tune-algorithm-parameters-with-scikit-learn/