CSE472 - Fake News Detection Using NLP Techniques and Machine Learning

Tara Paranjpe
Arizona State University
tparanjp@asu.edu

Stephanie Lee
Arizona State University
slee454@asu.edu

I. Abstract

Due to the amount of fake news circulating all over the media, fake news detection can be used to verify the credibility of these claims. By scraping the fact check websites in the train dataset and getting the term frequency of a true-false list in return, we were able to train the model with the Logistic Regression classification model and obtain an accuracy result from label predictions with Mean Squared Error metric.

II. Introduction

With how easily news and information can be obtained, a large amount of fake news and misinformation is bound to spread. The spread of fake news negatively affects our community in ways that obstruct the authenticity of the news community, deliberately provides falsified information, and rapidly changes how people perceive and respond to news. In order to verify the claims of information, a fake news detector is used to determine whether or not the claims are truthful in nature.

Given both train and test datasets, the fake news detector learns from the provided train dataset and predicts the authenticity of the claims in the test dataset. The goal of the fake news detector is to correctly classify the predicted label from each claim with Mean Square Error as a metric. Each label classifies the claims into one of the 4 categories:

1.) False (0): claim is proven false with a fact checker
2.) Misleading (1): claim is written in a confusing way that misleads people
3.) True (2): claim is proven true with a fact check
4.) Unproven (3): claim is not proven by fact check to be fake or true news

Solution Overview

The solution to our model implementation was to scrape the fact check websites using Selenium and BeautifulSoup libraries. We then used term frequency on specific keywords in lists. Next, we took the frequency counts, along with the corresponding label for each claim, and trained the model and fit it with the Logistic Regression classification model in order to receive the prediction for test labels.

III. Related Works

For our model implementation and experimentations, the following api and libraries were used and their purpose explained in respect to how they were used for this project:

1.) Selenium [6] : Webdriver used to visit websites for web scraping purposes
2.) BeautifulSoup [8]: HTML and XML file parser used for pulling data from websites.
3.) Sklearn [5] and [9]: Machine learning and data analysis library used for training and testing dataset and classifying

4.) Translate-api [13]: Translation api that provided different web translators as a tool to translate texts
5.) Afinn [2]: NLP tool for scoring texts with Sentiment Analysis
6.) Pandas [10]: A library used for data manipulation

IV. Model Description

For our final submission, the input features we utilized took the term count of a truth list and false list for each body of the article. The finalized model used to train and predict a classification was the Logistic Regression model with a ‘liblinear’ solver and regularization strength set to 0.01.

Feature Extraction

Our input features were chosen after meticulous experimentation and analysis of the data and models. We created 2 lists with common “true” and “false” synonyms. We pulled the contents of the fact check article provided in the dataset using the Beautiful Soup package and removed the “style” and “script” tags. We then utilized the Translators package to translate the contents of the article to English. Then, we used the Countvectorizer from SkLearn to check for the frequency of terms from the true and false lists. The frequency of true and the frequency of false were stored as separate terms in the feature vector.

As we looked further into other features (experiments defined in “Experiment” section), we struggled with the time intensity of the Selenium web driver and the consequent article scraping analysis. Appending features together to add to the feature vector characteristics proved to be even more difficult when we implemented logic to remove the row of training data when an error occurred. Because of this design decision that was made at the early stages of our project, we were unable to actually append other features since we did not keep track of the article associated with the row.

We realized this feature extraction was not the best way to attack this problem, and we complemented our research with other feature extractions. However, we were not able to combine these features due to earlier mentioned design decisions and the time intensity of the data pre-processing.

Model Selection

Among all of the models we experimented with, we settled on using Logistic Regression for our final submission. Logistic Regression is a classification model, and is simple to use as it is linearly separable. Logistic Regression is developed with class probabilities in mind, and provides a direction of association, such as positive or negative orientation. Within our model, we made use of some hyperparameters provided by Sklearn. We changed the solver of the model to LibLinear. This choice was made because it performs well with small datasets, like the dataset we collected. We also altered the regularization strength (“C” parameter) to 0.01. Regularization strength reduces the penalty against complexity, and helps reduce noise within data points given to the model, which can improve the model performance and accuracy.

We also experimented with different types of models, such as Multinomial Naive Bayes, which is helpful for sentiment analysis and probabilistic inputs, and K-Nearest Neighbors, which provides a clustering architecture for classification of data.

V. Experiment

Over the course of this project, we performed several different experiments to train our model and evaluate the results from the training data

Experiment 1: Term Frequency with True False Lists

To begin, we manually looked through some of the fact check articles, and tried to derive a pattern for the verdict in the article. We noticed from the articles that several of them contained keywords which resolved the verdict. We utilized Beautiful Soup and other HTML processing algorithms to get the article contents. Since there were several articles written in different languages, we normalized the data by translating the text to english. We then used the CountVectorizer provided by Sklearn to generate a term frequency of words in the article. We initialized two lists, a “true” list and a “false” list, as shown below:

True List : [“true”, “truth”]
False List: [“false”, “fake”]

We then passed this data as values in a data frame, with headers “truthcount” and “falsecount”, along with the expected label for the training data. Using a Logistic Regression model with a solver set to “liblinear” and regularization strength of 0.01, we trained our model and used the same features to predict the test data.

We realize this is not the best approach, nor is it representative of the whole article, as there are several chances for false interpretation of terms, like “not true”, “not false, and more. From this exercise, however, we were able to learn and observe other features to extract and experiment with.

Mean Square Error Scores for This approach

KNN Model: 1.23960
KNN Model with Error Value Set to 0: 1.05
Logistic Regression: 0.5874
Logistic Regression with more training data: 0.60
Random Forest Classifier: 0.6618
Random Forest Classifier with more training data: 0.699
Logistic Regression solver = ‘liblinear’: 0.58
Logistic Regression solver = ‘liblinear’ and C = 0.1: 0.5528
Logistic Regression solver = ‘liblinear’ and C = 0.01: 0.53733

Experiment 2: Sentiment Analysis on Fact Check Article Titles

The next experiment consisted of applying sentiment analysis on the titles of the fact check articles provided. Explained in [2], the Afinn Sentiment Analysis Python package has the ability to calculate how negative or positive a statement is. Our hope was that the Afinn package could take the fact check articles, and see if the article was more negative. For example, for a title like, “No, Kamala Harris did not say X”, Afinn could calculate a more negative sentiment, indicating to our model that the claim was more false leaning. However, our experiment quickly fell short when we ran across an article that dealt with a more negative topic, such as “Yes, COVID-19 caused terrible, agonizing deaths”, which had a very negative connotation, but was actually true in nature.

We struggled to test sentiment analysis on the whole body of the article, as we didn’t know if this would provide promising results, and it was very time intensive.

Mean Square Error Scores for This approach

No models tested with this approach due to lack of strength in approach.

Experiment 3: Analysis of Source of Claim

At this stage, we looked into leveraging some readily available data in the csvs rather than using the WebScraper logic from Selenium. The source type, such as “facebook”, “social media”, and others, was chosen as we observed that claims made on social media platforms versus news articles, for example, tended to have a tangible “positive-negative” correlation. In our input to the model, we leveraged “one hot encoding” to preserve the source type in a binary format to avoid skewing weight measurements in our model.

Extracting the source of each article for our feature vector proved to be less time-intensive and quicker for our model to train. We used a Multinomial Naive Bayes Model to test this. Multinomial Naive Bayes is a classification model, and is simple to use with a large dataset and feature vector. The common uses for this type of model include frequency calculations, and can more accurately predict multiclass problems compared to others. Within our model, we made use of some hyperparameters provided by

Sklearn. We altered the smoothing parameter of the model to 0.5. This value in smootion helps deal with the issue of zero probability of specific values within a model. However, this proved to be efficient, as we achieved a better least mean square error score, but realized to improve accuracy, we would have to append more features.

Mean Square Error Scores for This approach

Multinomial Naive Bayes, Alpha = 0.5: 0.52541

Experiment 4: Term Frequency of the Claim

Finally, as a last test before settling on our final solution, we looked at evaluating the term frequency of the claim provided in the dataset using the tfidfvectorizer. We discovered the ngram count that could look for ‘n’ number of words together, which proved to be helpful in tackling our earlier issue (detailed in “Experiment 1” above). We created a new truth list as follows:

True List : [“true”, “truth”, “real”, “accurate”, “correct”, “not false”, “not fake”]
False List: [“false”, “fake”, “wrong”, “inaccurate”, “not true”, “not right”]

However, this quickly proved to not lead anywhere, as the claim rarely has a mention of these terms. As we thought of this approach toward the end of the project, we didn’t have time to run this method on the whole body of the article, or the title of the fact check article.

Mean Square Error Scores for This approach

No models tested with this approach due to lack of strength in approach.

Through this exercise, we decided to go with Experiment 1 for our final submission. We were able to understand why the code worked, and justify the Logistic Regression model and tune the hyperparameters using the model specifications on the Sklearn website in [5].

Through these various experiments related to this Fake News Detection project, we were able to learn the importance of allowing ample time for pre-processing and feature extraction. More of our learnings are detailed in Section VII.

VI. Future Works

With more adequate knowledge from our research to implement a fake news detector, we have several ideas for how to proceed individually from the course after this project.

Append Several Features Together

One of the big issues we had during this project was time. When we realized that we had some data that couldn’t be combined with other data points, it was too late to combine features and create a new feature vector due to the speed of the Selenium Web Scraper. After this project, it would be fun to experiment how the model’s accuracy changes when more features are added to the vector and passed into the model. Some features we would try to add would include the source of the claim, and maybe attempt Sentiment Analysis and other NLP techniques.

Learn other NLP Techniques

Through this project, we learned way more about NLP than we had ever before, and got to actually implement NLP techniques, which is something we had never done before. With extra time, we would like to explore more NLP techniques, and understand how the techniques are used in other practical applications.

More robust web scraping techniques

One of the problems we faced for our project implementation is proper web scraping. We used BeautifulSoup to access the contents from each fact check website. But since each website structured their HTML differently, it became quite difficult to extract the correct information. We need to further research on the different techniques used with BeautifulSoup and apply the most efficient one when applicable. There are most likely other web scraping libraries available that we can experiment with. We believe that learning this technique properly can be helpful for future works that involve text analysis.

VII. Learnings

We have learned a lot about machine learning, data processing, inputs and outputs of machine learning models, and the importance of understanding your data before trying to pre-process it. We both have a large interest in Natural Language Processing (NLP), and this was a great project to explore how NLP works, since neither of us had that prior knowledge.

When we began the project, we researched quite a lot on how to get started with machine learning. For learning purposes, we stumbled upon a Medium article by Manthan Bhikadiya [1] that breaks down the steps to implementing fake news detection using machine learning. He explained the use of tfidf vectorizer to determine the relevance of words across all the news articles and getting a sparse matrix of terms as a result. By utilizing the matrix with a classification model, it is possible to train the model and fit using Multinomial Naive Bayes to get an accurate score. This fake news detection demonstration differs from ours in such a way that they did not have to scrape the fact check website for content. The dataset used for their model includes features such as the article title, article content, and label. This dataset is easier to work with, due to the fact that no web scraping is necessary and collecting the term frequency is much faster without accessing each fact check website. We decided to go a different route and implement a different term frequency instead of using tfidf vectorization and experiment with different classification models. But nevertheless, the article provided us insights to a better understanding of machine learning using python.

We also learned the importance of using the time given, and understanding and allowing for ample time in case an experiment provides unexpected results or data processing proves to be time intensive.

Through this project, we enjoyed being able to apply some of the concepts we learned in class to a real life problem, and hope that this experience and project will help us better tackle real-life situations related to machine learning and data mining.

VIII. References

[1] M. Bhikadiya, “Fake News Detection Using Machine Learning,” The Startup, Mar. 28, 2021. https://medium.com/swlh/fake-news-detection-using-machine-learning-69ff9050351f

[2] “Python - Sentiment Analysis using Affin,” *GeeksforGeeks*, Nov. 25, 2020. https://www.geeksforgeeks.org/python-sentiment-analysis-using-affin/

[3] A. Bandi, “Web Scraping Using Selenium — Python,” Medium, Jan. 22, 2019. https://towardsdatascience.com/web-scraping-using-selenium-python-8a60f4cf40ab.

[4] “TF - IDF for Bigrams & Trigrams,” GeeksforGeeks, Sep. 23, 2019. https://www.geeksforgeeks.org/tf-idf-for-bigrams-trigrams/

[5] Sklearn Logistic Regression Classification Model https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[6] “SeleniumHQ Browser Automation,” www.selenium.dev. https://www.selenium.dev/.

[7] “What Is Naive Bayes Algorithm In Machine Learning? | Analytics Steps,” analyticssteps.com. https://www.analyticssteps.com/blogs/what-naive-bayes-algorithm-machine-learning.

[8] “Beautiful Soup Documentation — Beautiful Soup 4.4.0 documentation,” Crummy.com, 2019. https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

[9] sklearn.feature_extraction.text.CountVectorizer — scikit-learn 0.20.3 documentation," Scikit-learn.org, 2018. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html.

[10] "pandas.DataFrame — pandas 1.2.4 documentation," pandas.pydata.org. https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html.

[11] "5 most effective neuro linguistic programming techniques," tonyrobbins.com, Feb. 12, 2021. https://www.tonyrobbins.com/leadership-impact/nlp-techniques/.

[12] "Advantages and Disadvantages of Logistic Regression," GeeksforGeeks, Aug. 25, 2020. https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/.

[13] UlionTse, "translators: Translators is a library which aims to bring free, multiple, enjoyable translation to individuals and students in Python.," PyPI. https://pypi.org/project/translators/

[14] "How to Find XPath in Chrome Browser," Scientech Easy, Jul. 25, 2020. https://www.scientecheasy.com/2020/07/find-xpath-chrome.html/

[15] "How to build a Web Scraper with Python and Selenium," WebScrapingAPI, Jul. 06, 2021. https://www.webscrapingapi.com/python-selenium-web-scraper/

[16] Neeraja Vaidya, "5 Natural Language Processing Techniques for Extracting Information," Aureusanalytics.com, 2015. https://blog.aureusanalytics.com/blog/5-natural-language-processing-techniques-for-extracting-information.

[17] DataFlair Team, "Train and Test Set in Python Machine Learning - How to Split - DataFlair," DataFlair, Aug. 06, 2018. https://data-flair.training/blogs/train-test-set-in-python-ml/.

[18] "1. Supervised learning — scikit-learn 0.22 documentation," Scikit-learn.org, 2019. https://scikit-learn.org/stable/supervised_learning.html.

[19] "sklearn.metrics.mean_squared_error — scikit-learn 0.24.2 documentation," scikit-learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html.

[20] "Python | Mean Squared Error - GeeksforGeeks," GeeksforGeeks, Jun. 28, 2019. https://www.geeksforgeeks.org/python-mean-squared-error/.