Fake News Detection

Faisal Ahmed
Department of Computer Science & Engineering
East West University
Dhaka, Bangladesh
faisal95bd@gmail.com

Asifuzzaman Miah
Department of Computer Science & Engineering
East West University
Dhaka, Bangladesh
zamanasif71.za@gmail.com

Thajiba Tabassum
Department of Computer Science & Engineering
East West University
Dhaka, Bangladesh
thajibatabassum@gmail.com

***Abstract*—Fake news is a phenomenon which is having a significant impact on our social life, in particular in the political world. Fake news detection is an emerging research area which is gaining interest but involved some challenges due to the limited amount of resources (i.e., datasets, published literature) available. We propose in this paper, a fake news detection model that use natural language processing and machine learning techniques. Moreover, majority of the existing fake news detection models treat the problem at hand as a binary classification task, which limits model's ability to understand how related or unrelated the reported news is when compared to the real news. To address these gaps, we present neural network architecture to accurately predict the stance between a given pair of headline and article body. We are able to achieve an accuracy of 93.7% on test data. We also discuss related research areas, open problems, and future research directions for fake news detection.**

Keywords—Fake News, Detection, NLP, Attack, Appears, Fact Checking, Model

I. Introduction (*Heading 1*)

"Fake News" is a term used to represent fabricated news or propaganda comprising misinformation communicated through traditional media channels like print, and television as well as non-traditional media channels like social media. The general motive to spread such news is to mislead the readers, damage reputation of any entity, or to gain from sensationalism. It is seen as one of the greatest threats to democracy, free debate, and the Western order [2]. Fake news is increasingly being shared via social media platforms like Twitter and Facebook [1]. These platforms offer a setting for the general population to share their opinions and views in a raw and un-edited fashion. Some news articles hosted or shared on the social media platforms have more views compared to direct views from the media outlets' platform. Research that studied the velocity of fake news concluded that tweets containing false information reach people on Twitter six times faster than truthful tweets [2]. The adverse effects of inaccurate news range from making people believe that Hillary Clinton had an alien baby, trying to convince readers that President Trump is trying to abolish first amendment to mob killings in India due to a false rumor propagated in WhatsApp.

First Draft News, an organization dedicated to improving skills and standards in the reporting and sharing of online information, has recently published a great article that explains the fake news environment and proposes 7 types of fake content: [3]

1. False Connection: Headlines, visuals or captions don't support the content
2. False Context: Genuine content is shared with false contextual information
3. Manipulated content: Genuine information or imagery is manipulated
4. Satire or Parody: No intention to cause harm but potential to fool
5. Misleading Content: Misleading use of information to frame an issue/individual
6. Imposter Content: Impersonation of genuine sources
7. Fabricated content: New content that is 100% false

This is the best and most complete categorization that I have seen and I know that a lot of work and research have been devoted to compile it.

Technologies such as Artificial Intelligence (AI) and Natural Language Processing (NLP) tools offer great promise for researchers to build systems which could automatically detect fake news. However, detecting fake news is a challenging task to accomplish as it requires models to summarize the news and compare it to the actual news in order to classify it as fake. Moreover, the task of comparing proposed news with the original news itself is a daunting task as its highly subjective and opinionated.

II. Background Study

Recent years, fake news (or rumor, misinformation) detection on social media has gained particular attention in the literature. A major track of existing studies aims at developing machine learning-based classifiers to automatically determine whether a news story spreading in a social media environment is fake based on a variety of news characteristics. A few early studies try to detect fake news based on linguistic features extracted from the text content of news stories.
Other methods have included:
Play around with a new dataset, test out some NLP classification models and introspect how successful they were?
Defining fake news with simple bag-of-words or TF-IDF.

Ultimately, there is still much work to be done within the field to advance the work toward a well functioning model for detection.

III. METHODOLOGY

People read texts. The texts consist of sentences and also sentences consist of words. Human beings can understand linguistic structures and their meanings easily, but machines are not successful enough on natural language comprehension yet. So, we try to teach some languages to machines like we do for an elementary school kid. This is the main concept; words are basic, meaningful elements with the ability to represent a different meaning when they are in a sentence. By this point, we keep in mind that sometimes word groups provide more benefits than only one word when explaining the meaning. Here is our sentence "I read a book about the history of America."

The machine wants to get the meaning of the sentence by separating it into small pieces. How should it do that? It can regard words one by one. This is ***unigram***; each word is a gram.

"I", "read", "a", "book", "about", "the", "history", "of", "America"

Introduction to Natural Language Processing:

Our news article data and headlines are in text format. Building automated machine learning models on text data involves cleaning and converting textual data to machine readable format. Natural-language processing (NLP) is an area of computer science and artificial intelligence concerned with the interactions between computers and human (natural) languages, and how to program computers to fruitfully process large amounts of natural language data [4]. In this project, we are leveraging textual data from FNC-1.

Natural language processing (NLP) models have made significant progress in automatically detecting sentiment and in mining opinion from text. A wide variety of benchmark datasets and techniques have emerged due to significant research conducted in this space [5]. Most of the pre-neural network era NLP techniques focused on developing extensive domain specific features. These techniques often involved manual feature engineering and they require linguists and semantic experts to parse and curate the text. However, the NLP landscape has evolved at great pace. Colbert et al. (2011) [6] introduced Natural Language Processing nearly from scratch which introduces unified neural network architecture and algorithms that can be applied to various NLP tasks. This paper [6] describes how we can learn word and sentence level representations instead of exploiting man-made input features carefully optimized for each task. This breakthrough research paved way for researchers to represent words and sentences as vectors, which can understand the context in which they are being used.

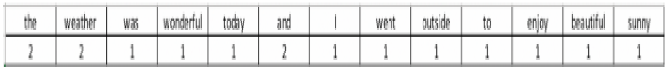
Word Vector Representation:

Natural Language Processing allows us to convert the text we want to analyze and preprocess them into features to be put into our model.

There are methods that can be used in the Natural Language Processing toolbox that can be used to preprocess our text: Count Vectorization and Term Frequency-Inverse Document Frequency. [7]

Count Vectorization involves counting the number of occurrences each words appears in a document (i.e distinct text such as an article, book, even a paragraph!). Python's Sci-kit learn library has a tool called CountVectorizer to accomplish this.

Example sentence: "The weather was wonderful today and I went outside to enjoy the beautiful and sunny weather." You can tell from the output below that the words "the", "weather", "and "and" appeared twice while other words appeared once. That is what Count Vectorization accomplishes.



the	weather	was	wonderful	today	and	I	went	outside	to	enjoy	beautiful	sunny
2	2	1	1	1	2	1	1	1	1	1	1	1

If you want to play around with Count Vectorizer, you can fiddle with three parameters. First of them is max_df. This pretty much sets how many features, or in this case words, you want Count Vectorizer to count. Setting this parameter can be incredibly useful when dealing with a large amount of documents as you'll run into speed issues with computation. Another is ngram_range. This deals with contiguous sequences of words. Let's say you want to deal with features such as "peanut butter and jelly", "intercontinental flight", or "rush hour traffic", you can get a lot more meaning than if you put all of these words as independent features. You could lose the meaning if you don't set this parameter. Also, what if you want to remove common words that could be irrelevant or unnecessary? Stop words! Stop words are common words (i.e. English stop words such as a, of, and, the) that can be removed in order to focus on more relevant words in your analysis.

But..

Knowing just when a word occurs in documents when they appear very frequently or infrequently across documents isn't very helpful. But, whether a word appear frequently in some documents but less frequently in others can be really useful! This is where **Term Frequency-Inverse Document Frequency (TF-IDF)** comes in!

- The term frequency refers to how much a term (i.e. a word) appears in a document
- Inverse document frequency refers to how common or rare a term appears in a document.

Inverse document frequency takes the logarithmic function of the size of the set of documents, D, and in how many documents a word appears even if it appears more than once within a document. This is then multiplied by the term frequency to get a score.

If the TF-IDF score is pretty high, it means the words is pretty rare and is good at discriminating between documents. This can be very useful unlike CountVectorizer only counts how many times a word occurs. If your analysis wants to look at the differences between documents, the uniqueness of words then TF-IDF is the tool to use. If you care how frequently a word appears, CountVectorizer is the tool to use. Again, it all depends on what you want to analyze. You have to understand your text to know what tools you can use. You can certainly use a combination of these techniques along with others to address your problem statement.

Our Dataset:

Online news can be collected from different sources, such as news agency homepages, search engines, and social me-dia websites. However, we collected our dataset from kaggle.[8]

This dataset have : 4 columns

#
title
text
label

title column has 6256 unique values.
text column has 6060 unique values.
Label column has 50% real and 50% fake.

Procedures:

Data Exploration:
To begin, you should always take a quick look at the data and get a feel for its contents. To do so, use a Pandas DataFrame and check the shape, head and apply any necessary transformations.

Extracting the training data:
Now that the DataFrame looks closer to what you need, you want to separate the labels and set up training and test datasets.
For this notebook, I decided to focus on using the longer article text. Because I knew I would be using bag-of-words and Term Frequency–Inverse Document Frequency (TF-IDF) to extract features, this seemed like a good choice. Using longer text will hopefully allow for distinct words and features for my real and fake news data.

Building Vectorizer Classifiers:
Now that you have your training and testing data, you can build your classifiers. To get a good idea if the words and tokens in the articles had a significant impact on whether the news was fake or real, you begin by using CountVectorizer and TfidfVectorizer.

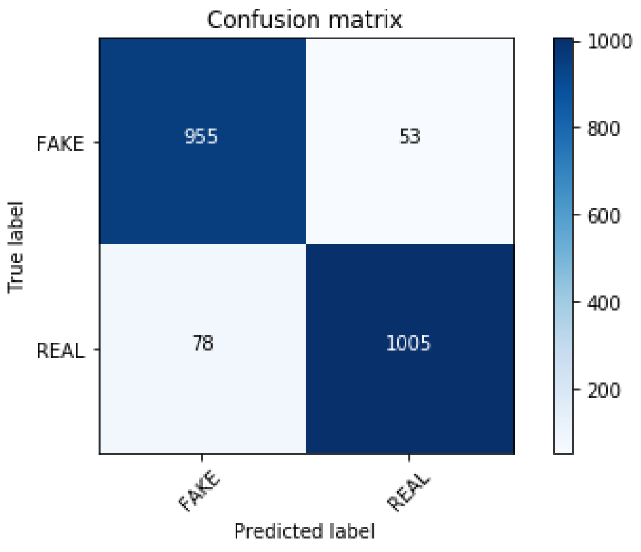
You'll see the example has a max threshhold set at .7 for the TF-IDF vectorizer tfidf_vectorizer using the max_df argument. This removes words which appear in more than 70% of the articles. Also, the built-in stop_words parameter will remove English stop words from the data before making vectors.
There are many more parameters avialable and you can read all about them in the scikit-learn documentation for TfidfVectorizer and CountVectorizer.

Testing Linear Models:
accuracy: 0.937
Wow!
I'm impressed. The confusion matrix looks different and the model classifies our fake news a bit better.
I personally find Confusion Matrices easier to compare and read, so I used the scikit-learn documentation to build some easily-readable confusion matrices (thanks open source!). A confusion matrix shows the proper labels on the main diagonal (top left to bottom right). The other cells show the incorrect labels, often referred to as false positives or false negatives. Depending on your problem, one of these might be more significant. For example, for the fake news problem, is it more important that we don't label real news articles as fake news? If so, we might want to eventually weight our accuracy score to better reflect this concern.
Other than Confusion Matrices, scikit-learn comes with many ways to visualize and compare your models.

Evaluation Metrics
To evaluate the performance of algorithms for fake news detection problem, various evaluation metrics have been used. In this subsection, we review the most widely used metrics for fake news detection. Most existing approaches consider the fake news problem as a classification problem that predicts whether a news article is fake or not.

• True Positive (TP): when predicted fake news pieces are actually annotated as fake news;
•True Negative (TN): when predicted true news pieces are actually annotated as true news;
•False Negative (FN): when predicted true news pieces are actually annotated as fake news;
•False Positive (FP): when predicted fake news pieces are actually annotated as true news.

Accuracy = |T P|+|T N| / |T P|+|T N|+|F P|+|F N|



IV: FUTURE WORKS
We tried unigram; each word is a gram.
In future we can try:
1. bigram (digram): each two adjacent words create a bigram.
"I read", "read a", "a book", "book about", "about the", "the history", "history of", "of America"
2. trigram: each three adjacent words create a trigram. "I read a", "read a book", "a book about", "book about the", "about the history", "the history of", "history of America"
3. We can add Artificial intelligence. To fasten and more accurate our project.

V:.CONCLUSION

So was your fake news classifier experiment a success? Definitely not.
But we did get to play around with a new dataset, test out some NLP classification models and introspect how successful they were? Yes.
As expected from the outset, defining fake news with simple bag-of-words or TF-IDF vectors is an oversimplified approach. Especially with a multilingual dataset full of noisy tokens. If we hadn't taken a look at what your model had actually learned, we might have thought the model learned something meaningful. So, we have to remember: always introspect your models (as best we can!).

.

References

[1] Allcott, H., Gentzkow, M.: Social media and fake news in the 2016 election. Technical report, National Bureau of Economic Research (2017)

[2] Langin, K.: http://www.sciencemag.org. Fakenews spreads faster than true news on Twitter—thanks to people, not bots(2018)J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] How can Machine Learning and AI help solving the Fake News Problem: https://miguelmalvarez.com/2017/03/23/how-can-machine-learning-and-ai-help-solving-the-fake-news-problem/

[4] https://en.wikipedia.org/wiki/Natural-language_processing

[5] Pang, B., Lee, L.: 2008. Opinion mining and sentiment analysis.http://www.cs.cornell.edu/home/llee/omsa/omsa.pdf

[6] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, M., Kuksa, P.: 2011. Natural Language Processing (Almost) from Scratch. Journal of Machine Learning Research 12 (2011) 2493-2537.

[7] Natural Language Processing: Count Vectorization and Term Frequency — Inverse Document Frequency: https://medium.com/@joshsungasong/natural-language-processing-count-vectorization-and-term-frequency-inverse-document-frequency-49d2156552c1

[8] https://www.kaggle.com/rchitic17/real-or-fake