Fake News: An Adversarial Training Approach

Natalia Valencia

Abstract

The current era of information communication provides various types of information instantly and on various different platforms i.e. Television, Twitter, Facebook, etc. While this is incredibly useful, it also means there has been a large influx of dissemination of misinformation, by in large "Fake News". Although the aim of this paper is to improve fake news classification models using adversarial training, thus, improving model robustness, it is not always the case that the models produce the desired results. I employ the use of sentence negation to train my Fake News Classification model.

1 Introduction

The internet has become a hotbed for the dissemination of fake news. Most of us focus primarily on the headlines instead of paying attention to the content of the news. This can lead us to accept fake news more readily at face value. This would not be so bad if the dissemination of fake news was limited to a finite amount of places, but since the internet spreads information like wildfire, it's hard to control the flow of incoming news. With the widespread use of social media, fake news is everywhere. This is concerning because it causes a lot of distress and inconvenience to people's lives. When Iranians were protesting against the government, a few reputable sources claimed that Iranian law decreed the mass execution of protesters. This news infuriated many public figures, one being the Canadian prime minister, who proceeded to decry the Iranian government on Twitter for committing such a heinous act. (McQuillan, 2022) Not only was Trudeau fooled into disseminating fake news, but he made a blunder, a blunder that does not reflect well on his role as Prime Minister.

Much research has been done on automating fake news classification. Fake news detection on social media has been looked at from a data mining perspective(Shu et al., 2017). Machine learning algorithms have been used in detecting fake news (Khan et al., 2019). ML classifiers that have been used for fake news classification include SVM, Naïve Bayes classifiers, decision trees, and gradient enhancement, to name a few (S. Gilda, 2017). Most recently, deep learning algorithms have shown better accuracy and performance when dealing with NLP tasks than basic ML algorithms. Deep learning algorithms, such as feed-forward networks, RNN, CNN, capsule networks, attention mechanisms, memory-augmented networks, graph neural networks, Siamese neural networks, and transformers, have proven useful in the task of text classification, with transformers making the biggest strides(Minaee et al., 2021).

2 Related Work

Recent research has focused primarily on making deep learning models more robust against adversarial attacks. (Szegedy et al., 2013) discovered that machine learning models, including neural networks, are vulnerable to adversarial attacks. Most adversarial attacks known to date, have been accomplished in the image and speech domains (Szegedy et al., 2013), (Carlini and Wagner, 2018). Due to the nature of text, it is still challenging to create adversarial examples. Previous works don't conform to the three key utility-preserving properties: human prediction consistency, semantic similarity, and language fluency (Jin et al., 2020). (Li et al., 2018) employ word misspelling, Li, Monroe, and (Jurafsky, 2016) use single-word erasure, and (Liang et al., 2017) incorporate phrase insertion and removal to create adversarial examples. Yet, these methods, usually result in unnatural sentences. (Jin et al., (2020) generate high-profile utility-preserving adversarial examples by applying a ranking to a word based on its importance, designating said word a replacement mechanism via synonym extraction, semantic similarity checking, and POS tagging, then selecting the highest

Negation Attack

Original Statement	*Modified Statement*
EU's Verhofstadt pokes fun at British PM	EU's Verhofstadt **doesn't** poke fun at British PM

Table 1: An example of the Negation adversarial attack.

semantic similarity word amongst the winning candidates.

3 Adversarial Training

Adversarial training with adversarial examples is, simply put, training a model with a combination of original and adversarial examples. Since we know that training in itself is susceptible to attacks, why not incorporate adversarial examples into the mix. Ideally, training a model with adversarial examples should reduce the likelihood of adversarial errors, but does not guarantee it will reduce adversarial errors when faced with different attacks.

In this paper, I consider a statement to be *fake* if it is factually incorrect, and *real* otherwise. Fake News detection models read text and base the validity on factors such as surrounding context and real-world facts. There are various type of attacks that can be used for training, but for this paper I will be focusing on a singular attack, Negation. The goal is to train the model using the attack vector to improve the rate of fake news detection accuracy. An example of what the negation attack looks like is shown in Table 1.

Before any training begins, a baseline must be drawn. For this I train the news classification model on the original data set to create the baseline for comparison. Next, I create the adversarial examples and incorporate them into my training data set. I then train the model on the new data set. Finally, I run the trained model on the test set to see how well it performs.

4 Negation Attack

Negation works by taking words and negating them to their opposites. Words such as "can" becomes "can not" and "is not" becomes "is". This process is automatically performed by a script and while the script is not guaranteed to negate 100 percent of the sentences, it does change enough to make an impact on the data. Ideally, a data set should contain a decent amount of adversarial examples for optimal training. For the purpose of this project, the current script results are enough to yield adequate training results.

5 GonzaolA / Fake News TFG Dataset

This data set repository is an English language data set containing over 45 thousand unique news articles. The articles have already been classified as either real (1) or fake (0). The structure is composed of 40,587 fields about news. These fields are: The title of the news, the text or content of the news, and the value of the news, whether the news is real (1) or fake (0). The data source is a mix of multiple fake news data sets from Kaggle.

6 Setup/Pipeline Overview

Pre-processing, Negation Pre-processing, Feature Extraction, Summary View of the Data, Algorithms, Model Architecture, Implementation, Results

6.1 Pre-Processing

In pre-processing I need to adjust the data to fit into my model. I begin by removing unnecessary data such as unused columns containing miscellaneous text and any unnamed or empty columns. The next setup is to convert all letters to lowercase. The next step is to impute all null values followed by punctuation removal. The final step is to remove stop words, however in this paper I will be performing tests with and without stop words to see how the trained models are affected. To remove stopwords I use NLTK's English stopwords.

6.2 Negation Pre-Processing

For the negation pre-processing I will be using the "Negate Python" module to negate sentences in the training and validation data sets. The Negate python model implements rule-based, syntactic sentence negation in English. The negate method will work for most cases but not for all. Support for the following list is not included:

- "Some", "any", and "yet"
- Inversions
  E.g., "Did you go to the concert?" vs. "You did go to the concert."

- Non-verbal negations
  "A bottle with no cap." will produce the warning: Negator - WARNING: Sentence not supported. Output might be arbitrary

- Auxiliary "ought"

- Certain verb conjunctions
  E.g.: "She hates and loves winter." → "She doesn't hate and love winter."

- Multiple verb negations
  E.g.: "I am hungry because I didn't eat." → "I am not hungry because I ate."

While this may seem like a lot of not supported negates, the majority of the English language is supported and will work for this use case.

On the other hand, the negate method I implemented only covered about 3,000 sentences out of the 24,000 + sentences. I felt that it would not produce as big of an impact as what I'd hope. Therefore, it is not used to conduct my experiments.

6.3 Feature extraction

For feature extraction, I will be using techniques such as, Word2vec embedding, Keras Tokenizer, and Keras Padding. The word2vec embedding model I use is "glove.6B.100d.txt" embedding file. The Glove embedding dictionary contains vectors for words in a 100 dimensions, mainly all words in the dictionary. Keras's Tokenizer allows me to take the text and convert it into tokens and the padding is used to equally distribute words in sentences, padding the remaining spaces with zeros. This padding is used to account for the difference in the size of words.

6.4 Summary View of the Data

This section includes bigrams for the top 20 most frequently occurring words for both Real and Fake News. The reason I'm including these images is two-fold: One, I want to show the inclusion/exclusion of stopwords and how that affects the sentence negation adversarial examples, and two, I want to provide a justification as to why I believe the model fluctuates. One of the reasons I believe the model does not provide accurate results has to do with the inclusion of "[", "]", ":", as well as other characters that remain unaccounted for.

Figure 1 shows the top 20 frequently occurring true news bigrams from the train dataset. As we can see, the set of words that appear the most in statements is (white,house), followed by (north,korea), and (trump,says). Figure 2 shows the top 20 frequently occurring fake news bigrams from the training dataset. The set of words that most frequently occur in statements happen to be stop words, and the list of verbs necessary for the negation attack to work. We can also see that both cases of the word video include extra characters. Figure 3, on the other hand, bumps (white,house) from its number 1 spot in Figure 1 to the third most frequently occurring true news bigrams once the negation is applied. This demonstrates how even something as simple as sentence negation can cause a significant change in how the dataset is presented. Figure 4 also demonstrates how perturbed the dataset becomes from Figure 2. The remaining 4 figures (5 - 8) represent the top 20 most frequent bigrams for both true and fake news with stopword removal and sentence negation. All four figures have characters that were not removed when I applied punctuation removal in the pre-processing step of the pipeline. Instances of video, factbox, and watch all have unaccounted for special characters. Video contains paranthesis and brackets, and both watch and factbox contain colons. With the removal of stop words, both true and fake news bigrams remain consistent, even after sentence negation. This just indicates that stop words removal destroys the attack vector.

6.5 Algorithms

The Negation module's negation method heuristically attempts to identify a sentence by getting a sentence's dependency parse, POS tags, and the morphological information of each word in the sentence as input. It then negates the sentence based on a set of rules.Each rule has a dependency tree regex pattern. Different Semgrex patterns are used to identify different syntactic templates, which then transforms a sentence based on an action defined by the rule. Actions include *move, replace, insert* and *lemmatize*.

The Negate method I created, but which is not used in the experiments, takes a sentence as input and returns its negation. The way it works is it proceeds to try to match every word in a sentence with a negate dictionary that covers the base cases: 'is: isn't', 'is: is not', 'isn't: is', 'is not: is', 'did: didn't', 'did: did not', 'didn't: did', 'did not:did', 'has: doesn't have', 'has: does

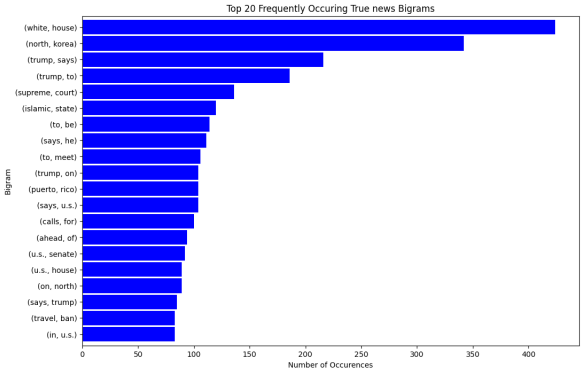

Figure 1: Training Data Bigram: True

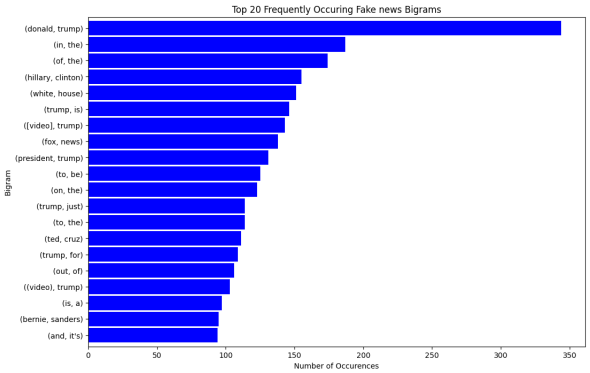

Figure 2: Training Data Bigram: False

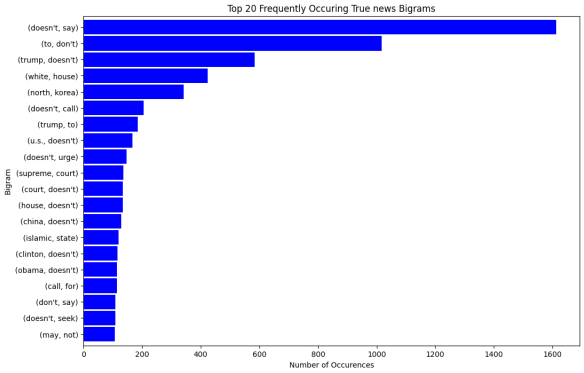

Figure 3: Negated Training Data: True

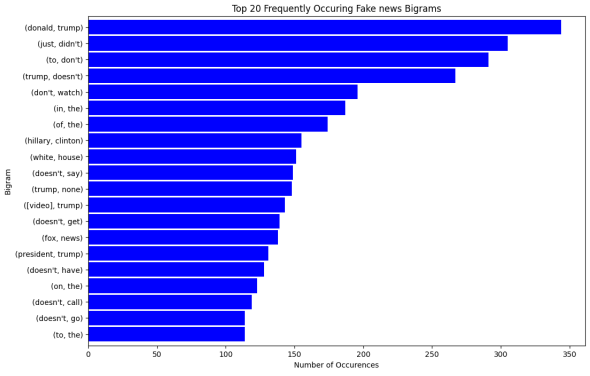

Figure 4: Negated Training Data: False

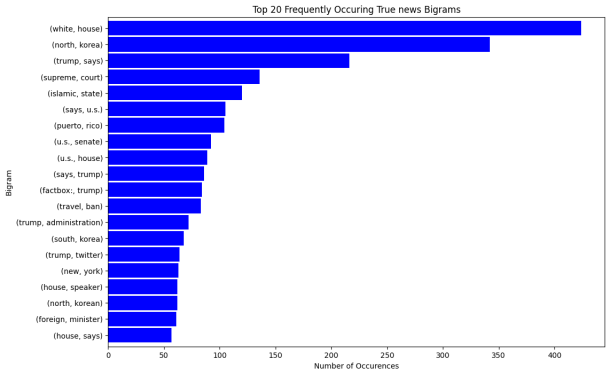

Figure 5: Training Data without stopwords Bigram: True

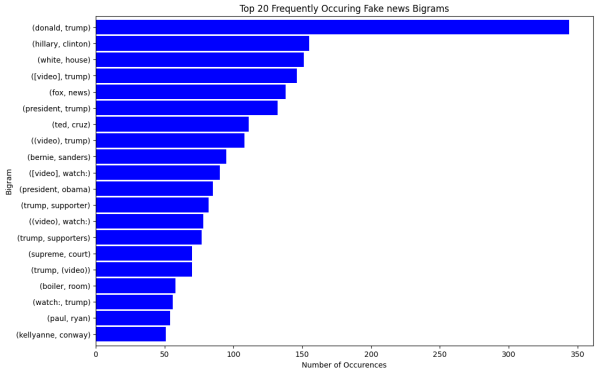

Figure 6: Training Data without stopwords Bigram: False

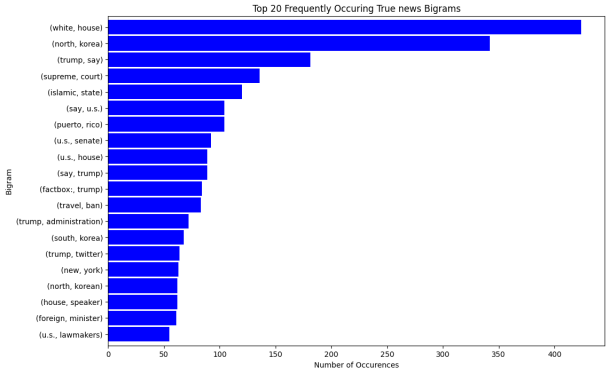

Figure 7: Negated Training Data without Stopwords Negated: True

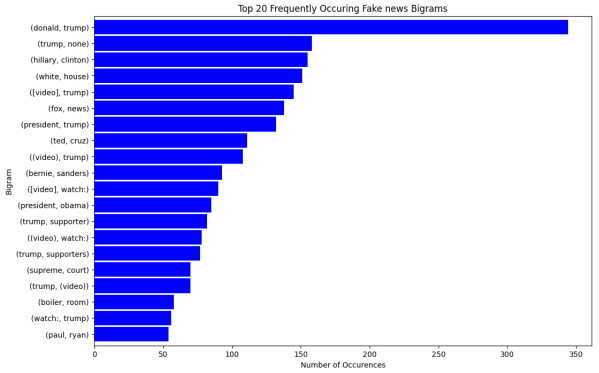

Figure 8: Negated Training Data without Stopwords Negated: False

not have', 'doesn't have: has', 'does not have: has', 'should: shouldn't', 'should: should not', 'shouldn't: should', 'should not: should', 'would: wouldn't', 'would: would not', 'wouldn't: would', 'would not: would', 'must: mustn't', 'must: must not', 'mustn't': must', 'must not: must', 'can: can't', 'can: cannot', 'can't: can', 'cannot: can'. If the word in the sentence matches the first word of each set of words, the method then proceeds to replace said word with the word or words that follow the colon. The method does account for other cases not present in the above dictionary. For words such as 'does' and 'doesn't/does not', I create a regex expression which can replace the original word so as to provide a wider range of negation, i.e., 'does work: doesn't/does not work'. Different verb types are also taken into account by implementing and if, else statement that takes several verb forms into consideration.

Other algorithms used, but not applied were name replacement and adverb intensity. My name replacement method utilized spacy to tokenize, parse, and tag the names of people in each sentence. I use a list to keep track of these entities and employ zip to create a set of names, which I can then replace with each other. My adverb intensity method switches polar words with one of the words from my booster dictionary.

The punctuation removal method I implement does not remove all character instances, it only removes punctuation. The method receives the text as input and creates a list for every character in string.punctuation. It then applies spaces to where the punctuation should be and joins them with the list. The method returns the cleaned list.

6.6 Model Architecture

The model architecture type is a dense layered sequential model. The model has a dropout layer set to 0.5 to reduce overfitting. I employ the use of a Sigmoid activation function because this is a binary classification problem, True (1) False (0). The Model is trained on the Adam optimizer with a 1e-4 learning rate. Relu activation is used in 4 of the 5 dense layers. I also track the binary cross entropy loss and train and test accuracy.
"All models are bad, some models are useful"

6.7 Implementation

For my experiments I made use of the following python libraries/modules:

- load dataset from datasets to load the HuggingFace dataset
- pandas to make dataframes out of the datasets
- string for punctuation removal
- spacy to tokenize,tag, and parse words
- shuffle from sklearn to randomize name replacement
- Negator from negate to negate the sentences in the dataset
- itertools to iterate through text
- nltk for removal of stop words
- Tokenizer from keras.preprocessing.text to tokenize the dataset
- pad sequences from keras.utils to add padding to sentences to equalize them
- numpy for matrix embedding
- tensorflow for deep learning
- matplotlib to visualize the results
- wordcloud to visualize word frequencies

My editor of choice was visual studio code with jupyter notebooks integrated into it. My python version is 3.11. I also made use of Google Colab's GPU environment for result replication, as well as, word cloud production. I did not use my laptop's nvidia GPU drive as its performance is slower than that of my CPU.

6.8 Results

The results are broken down into the following categories:

- Baseline

  The baseline results are probably the most accurate of the results. As we can see from Figure 9, the test accuracy in most cases performs better than the train accuracy. For the most part, there is no model overfitting.

- Adversarial Training

  The adversarial training results are concerning, as the test accuracy peaks and falls sporadically. Figure 10 indicates a few things: primarily, the test data is not reputable enough,

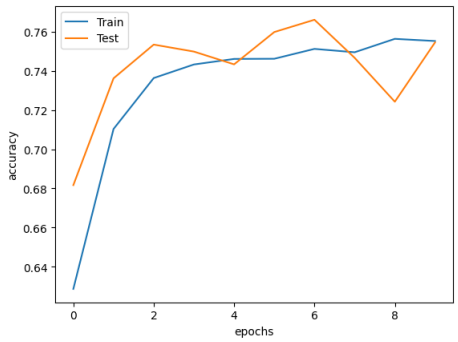

Figure 9: Baseline Results

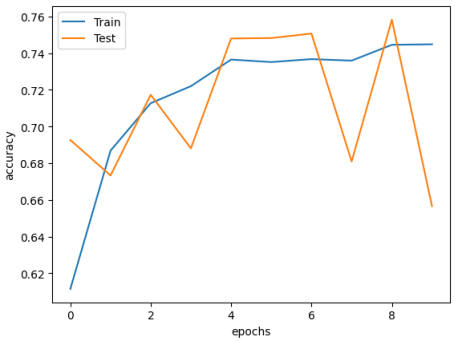

Figure 11: Adversarial Attack Results

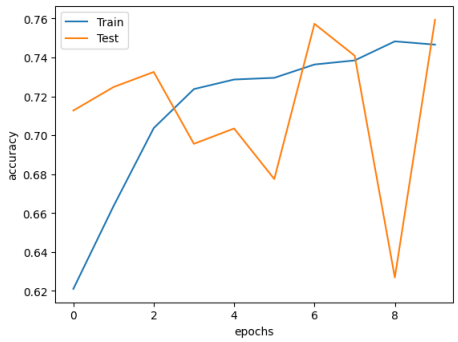

Figure 10: Adversarial Training Results

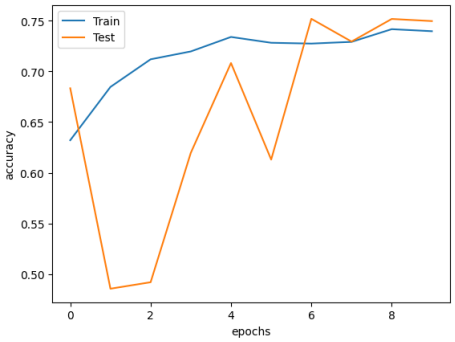

Figure 12: Adversarial Training/Attack Results

character inclusion is skewing the model accuracy, and/or evaluation metrics should focus on adversarial error loss instead of cross entropy loss and accuracy.

- Adversarial Attack

  An adversarial attack should yield a lower test accuracy. Our model does not produce the desired results. Figure 11 shows the test accuracy peaking, falling, and plateauing when ideally it should just perform worse than the training accuracy. Again, the list of factors that can be causing this fluctuation is non-exhaustive.

- Adversarial Training and Adversarial Attack

  One would assume that training a model on a combination of original and perturbed text and then attacking it would produce similar outputs for both the training and test accuracy. We can see from Figure 12 that that is not the case. The training accuracy definitely performs better, but the test accuracy is falling and plateauing. Perhaps accuracy was not the best measurement.

7 Discussion

It is clear to me that there is a strong need for further development in the study of fake news classifications. With the current level of understanding gathered from this paper, it can be seen that the code is still premature and has a long way to grow. There is potential for higher levels of accuracy across a larger range of data sets and training classification models with adversarial attacks could be a good method for achieving that goal. The code needs tuning and updating and different types of models and training attacks can be and should be used to train classification models. This paper is only an example of a single type of attack and training and there are a variety of different solutions available.

I believe there were several factors that affected the study and could be changed to yield improvements. First, the data set and its reliability. The data set was a mix from various different fake news data sets and came from different sources. This mix of data can skew results as different sources will label things in different ways. I believe it would be better to start training a model using single-source data sets and then move on to

different data sets to further increase accuracy.

Was the chosen model architecture wrong? The sequential model may not be the most powerful model out there but it is fairly accurate. I had a resource constraint and could not perform my experiments using a stronger model as I did not have the computing power necessary to train with higher-end models. It is possible that using the same methodology but with a different model would have been possible to achieve greater results.

During the training, I noticed that removing stop words hindered training and heavily skewed accuracy results. This again might also be a limitation of the model or potentially an issue with the data set. Further experimentation could give me more definitive results on if stop words affect all adversarial training or if they only affect specific models.

Other things to note include model fluctuations, the need for code improvement, using different word2vec embeddings, and better evaluation measurements all play a role in fake news classification. The lack of time and computing power hindered my ability to expand the research into all areas and further research is necessary.

Limitations

The two biggest limitations faced during this paper were time and compute resources. For a project of this nature, having only a few months of research and implementation is not enough to yield results that could create an immediate impact. This type of project requires hours and hours of research, programming, and testing to create something that would be deemed useful in the field. Alongside time, having proper compute power is a big limitation as training these models is a resource and time-consuming process. The more complex the code and model and data set the longer it will take to train and the more compute is necessary to run these calculations. I had to scale down my model due to this reason and even with a sequential, less powerful model it still took over 30 minutes per training to complete calculations, and I ran the programs multiple times per session. With enough time and compute it is possible to achieve a much higher accuracy rating using adversarial training.

7.1 References

CBC/Radio Canada. (2022, November 16). Don't let Trudeau's misinformation about Iran death sentences overshadow real abuses, experts warn | CBC news. CBCnews. Retrieved February 25, 2023, from https://www.cbc.ca/news/world/trudeau-tweet-iran-misinformation-1.6652749

Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. ACM SIGKDD Explorations Newsletter, 19(1), 22-36. https://doi.org/10.1145/3137597.3137600

Khan, J. Y., Khondaker, M., Islam, T., Iqbal, A., & Afroz, S. (2019). A benchmark study on machine learning methods for fake news detection. Computation and Language. https://arxiv.org/abs/1905.04749

Gilda, S. "Evaluating machine learning algorithms for fake news detection." 15th Student Conference on Research and Development (SCOReD) (pp. 110-115). IEEE. 2017

Akshay Jain and AmeyKasbe. "Fake News Detection." 2018 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS). Bhopal, India: IEEE. 2018.

Kaur, G., & Bajaj, K. (2016, May). News classification using Neural Networks - caeaccess.org. Retrieved February 27, 2023, from https://www.caeaccess.org/archives/volume5/number1/kaur-2016-cae-652224.pdf

Minaee, S., Kalchbrenner, N., Cambria, E., Nikzad, N., Chenaghlu, M., & Gao, J. (n.d.). Deep Learning Based Text Classification: A comprehensive review. Retrieved February 26, 2023, from http://export.arxiv.org/pdf/2004.03705

Kaur, G., & Bajaj, K. (2016, February). News classification and its techniques: A Review. Retrieved February 27, 2023, from https://www.researchgate.net/profile/Karan-Bajaj-2/publication/303501815_

News_Classification_using_Neural_Networks/links/5cec8e40299bf109da75064e/News-Classification-using-Neural-Networks.pdf?origin=publication_detail

Zhang, M. (2021, August 5). Applications of deep learning in news text classification. Scientific Programming. Retrieved February 26, 2023, from https://www.hindawi.com/journals/sp/2021/6095354/

Lee, D.-M., Kim, Y., & Seo, C. gyun. (n.d.). Context-based virtual adversarial training for text classification with noisy labels. ACL Anthology. Retrieved February 26, 2023, from https://aclanthology.org/2022.lrec-1.660/

TARIQ, A. B. D. U. L. L. A. H., MEHMOOD, A. B. I. D., ELHADEF, M. O. U. R. A. D., & KHAN, M. U. H. A. M. M. A. D. U. S. M. A. N. G. H. A. N. I. (2022, July). Adversarial training for fake news classification. Retrieved February 27, 2023, from https://www.researchgate.net/publication/362350845_Adversarial_Training_for_Fake_News_Classification/fulltext/635ca63412cbac6a3e055f58/Adversarial-Training-for-Fake-News-Classification.pdf

Ahmed, A., Aljarbouh, A., Donepudi, P., & Choi, M. (n.d.). Title: Detecting fake news using machine learning: A systematic ... Retrieved February 27, 2023, from https://arxivexport1.library.cornell.edu/pdf/2102.04458

Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017, September 3). Fake news detection on social media: A Data Mining Perspective. arXiv.org. Retrieved February 26, 2023, from https://arxiv.org/abs/1708.01967

Khanam, Z., Alwasel, B. N., Sirafi, H., & Rashid, M. (n.d.). Fake news detection using machine learning approaches - iopscience. Retrieved February 27, 2023, from https://iopscience.iop.org/article/10.1088/1757-899X/1099/1/012040

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015, March 20). Explaining and harnessing adversarial examples. [1412.6572] Explaining and Harnessing Adversarial Examples. Retrieved February 26, 2023, from http://export.arxiv.org/abs/1412.6572

Jin, D., Jin, Z., Zhou, J. T., & Szolovits, P. (2020, April 8). Is Bert really robust? A strong baseline for natural language attack on text classification and entailment. arXiv.org. Retrieved February 26, 2023, from https://arxiv.org/abs/1907.11932

Miyato, T., Dai, A. M., & Goodfellow, I. (2021, November 16). Adversarial training methods for semi-supervised text classification. arXiv.org. Retrieved February 26, 2023, from https://arxiv.org/abs/1605.07725

Gu, S., & Rigazio, L. (2015, April 9). Towards deep neural network architectures robust to adversarial examples. arXiv.org. Retrieved February 26, 2023, from https://arxiv.org/abs/1412.5068

Li, J.; Ji, S.; Du, T.; Li, B.; and Wang, T. 2018. Textbugger: Generating adversarial text against real-world applications. arXiv preprint arXiv:1812.05271

Li, J.; Monroe, W.; and Jurafsky, D. 2016. Understanding neural networks through representation erasure. arXiv preprint arXiv:1612.08220.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199.

Liang, B.; Li, H.; Su, M.; Bian, P.; Li, X.; and Shi, W. 2017. Deep text classification can be fooled. arXiv preprint arXiv:1704.08006.

Carlini, N., and Wagner, D. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. arXiv preprint arXiv:1801.01944.

Ethics Statement

All work conducted in this paper is performed with an ethical understanding of good and was not targeting any live news outlets or attempting to disseminate misinformation. All adversarial attacks were used for training a fake news classification model to increase the accuracy of finding and alerting to Fake news. No adversarial attacks were used in an unethical way such as lowering fake news detection in classification models. The author of this paper does not condone unethical behavior or the use of research to perform any type of unethical actions.