CLASSIFICATION OF FAKE NEWS

Axel Bogos - 40077502
G39
Gina Cody School of Engineering and Computer Science
Concordia University
Montreal, Quebec, CA
a_ogos@live.concordia.ca

December 8, 2020

ABSTRACT

Accurate classification of news article as legitimate or containing fake information has seldom been a more prominent issue. In this project, a survey of available datasets in news classification was conducted and an evaluation of common language models on the task of text classification was done. In addition to typical classification models like Random Forests and XGBoost, increasingly complex language models and worc embeddings such as Word2Vec and ELECTRA were compared and used to vectorize the input and classify articles. The ELECTRA and bi-directional LSTM models achieved the highest classification accuracy at 85%, however simpler models also showed reasonable performances.Among our conclusions, we also draw attention to some common issues in available datasets, some strengths and limitations of simpler word vectorization models like TFID and possible improvements combining simpler and state-of-the-art models.

1 Introduction

Discerning legitimate information from misleading or downright false content has never been a more ambiguous task. The proliferation of fake news has undermined the news ecosystem and the general public trust towards news outlet [1]. Moreover, the COVID-19 pandemic and the 2020 Presidential American Election has catalyzed this phenomena as both the quantity of fake news and the size of the potential audience has increased. While the ability to quickly communicate facts and policy to the population has rarely been as critical, it has been shown that fake news spread quicker and wider than true news [2]. Moreover, the constantly changing nature of fake news content presents particular challenges regarding dataset construction and model adaptability; for instance, a word such as "Chloroquine" may have been a strongly correlated with legitimate publications a year ago but its use-context has much changed since. Thus, robust, adaptable and highly scalable solutions are needed in order to evaluate and identify the veracity of news that are published across social media and news outlets. A number of studies have pursued similar goals [3][4][5]; in addition, methods of circumventing state-of-the-art fake news detection models have also been described[6]. This project does not intend on necessarily improving upon the former or defeating the latter. It rather aims at evaluating available datasets, quantifying the results of models onto the chosen dataset, reflect on their usability, identify potential improvements and hopefully provide simple yet useful models.

2 Data

Upon inspection of multiple datasets such as FakeNewsNet [7] and LIAR [8] some of their common limitations were:

1. Evaluating the ground truth of a claim in order to correctly label an article is non-obvious and possibly extremely labour intensive, leading to small datasets.
2. Most "Fact-Checking" agencies such as Politifact act within a certain domain of news and hence do not necessarily generalize well in order to provide a domain diverse dataset.

Table 1: 10 most common bigrams before and after pre-processing

Before	After
(of, the)	(united, state)
(in, the)	(donald, trump)
(to, the)	(new, york)
(on, the)	(white, house)
(for, the)	(president, trump)
(and, the)	(last, year)
(to, be)	(prime, minister)
(that, the)	(social, medium)
(at, the)	(trump, administration)
(in, a)	(last, week)

3. "Truths", particularly within the political domain, are very dynamic and subject to change. This makes it hard to keep up to date with words that might be re-appropriated from "real" news to "fake" news or vice-versa.

The chosen dataset NELA-2019 [9] addresses a number of these issues. It is organized as one JSON file per news publisher containing multiple articles and a label file where each news publisher receives multiple trustworthiness score from 0 to 2 (0 being the most reliable) as evaluated by the following news assessment groups: 1. Media Bias/Fact Check (MBFC) 2. Pew Research Center 3. Wikipedia 4. OpenSources 5. AllSides 6. BuzzFeed News 7. Politifact
Their respective evaluation are aggregated into a single reliability score per news providers. Only articles originating from news providers with an aggregated score of 0 or 2 were used in this project. By selecting the news providers at both ends of the reliability spectrum (in the sense that their articles reliability would be obvious to a critical human reader) and using their overall trustworthiness as a label article-wise, we are able to obtain a domain diverse dataset that is representative of both real and fake news content across more than 20 news providers. The relatively large number of news providers compared to the binary label "fake" or "true" reduces the possibility of us classifying the language patterns of a particular news provider instead of the language patterns proper to the class of the article
Overall, a dataset of 44,378 articles was created by sampling the NELA-2019 dataset where news providers are labeled as 0 or 2. They are sampled in a balanced manner with about 22,000 instances of each label. Each article instance is organized as shown in Table 3. An additional column "label" is created based on the source as such:

$$label = \begin{cases} 0 & \text{if source aggregated label = 2 (Fake)} \\ 1 & \text{if source aggregated label = 0 (True)} \end{cases}$$

3 Pre-processing

A new feature resulting from the concatenation of the title and body of the article was created and used as an input to models. This input was pre-processed using native Python functions and the Natural Language Toolkit (NLTK) package. An overview of the 10 most frequent bigrams in the input before and after pre-processing is shown in Table 1. The increase in meaningful words among the most frequent bigrams supports our choice of pre-processing methods, particularly stop-word removal and lemmatization.

3.1 Lowercasing and removal of non-alphanumeric characters

Every instance of the input was made lowercase using native Python String functions. Regular expressions were used to strip non-alphabetic characters, newline markers and punctuation from the input. Note that the input did not contain social media posts, hence no special treatment for Twitter handles or hashtag was necessary.

3.2 Removal of stop words

Stop words consist of words that are very common within a language but are generally devoid of strong meaning. Examples of such words in English might include "the", "to", "of", "a". While they might be useful for applications such as translation, we do not expect them to provide much insight into whether a news is fake or not. The NLTK library provides a widely used list of stop-words in multiple languages; its list of English stop words was used to filter the input.

Table 2: Modelling Results

Model	Accuracy	Precision	Recall	F1
Baseline (GNB)	0.79	0.80	0.79	0.79
Random Forest	0.82	0.82	0.82	0.82
XGBoost	0.83	0.83	0.83	0.83
LSTM	0.85	0.85	0.85	0.85
Electra	0.85	0.85	0.85	0.85

3.3 Lemmatization

Lemmatization and stemming are related processes that consist of mapping multiple words onto a single word encompassing their meaning. For example, $\{walk, walks, walking, walked\}$ may be mapped to the root word $\{walk\}$. Lemmatization differs from stemming in that it takes into account the context and the abstract meaning of a word, not only its radical form. For instance stemming may reduce $bettering$ to $better$, while lemmatization may go further and lemmatize it as $good$, hence further reducing the corpus. This may reduce training time and help extract language patterns and meaning. Thus, NLTK's WordNetLemmatizer was used on the input.

4 Modelling & Results

An overview of experimental results and metrics is shown in Table 2. Google Collabotory GPUs were used to train bidirectional-LSTM and ELECTRA models.

4.1 Vectorizers

A vectorized representation of text is a crucial part to any NLP process. Notably, Term Frequency Inverse Document (TFID) vectorization, count vectorization and Word2Vec strategies were used in this project. TFID vectorization uses the frequency of a word per document (an article) as well as its weight within the whole corpus (the vocabulary of all articles) as a numerical representation. This type of numerical transformation of an article does not take into account word order. While this may seem a significant simplification, it is a common assumption that reasonably successful models for text classification are possible by only examining words and not their relative ordering [10]. Count vectorization operates in a very similar way; however only the frequency of a word within a particular document is represented: they are not weighted across the corpus. Finally, Word2Vec is a pre-trained vectorization-more precisely a word-embedding- model which processes a corpus (the ensemble of words within a document) and outputs a vector representation of those words in a vector space. Word2Vec is pre-trained on large text corpus such as Wikipedia. GenSim Word2Vec library is commonly considered standard and was used to vectorize the input. Table 4 provides a few example of word similarity vectors as a sanity check. Overall, Count vectorizers were used to evaluate a baseline model, TFID for random forest and XGBoost models and Word2Vec was used in pair with an LSTM model. The vectorization is embedded in the ELECTRA model.

4.2 Baseline

A multinomial Naive-Gausssian Bayes classifier from the sklearn library was used as a baseline. The input was vectorized using a CountVectorizer and splitted as 80/20 train/test. As a baseline, the classifier was not further optimized and default hyper-parameters were used. The baseline classified the test set with a *0.79 accuracy rate* (see Table 2 for all metrics). Training was extremely fast with a 168ms wall time on CPU. The feature coefficients were examined in order to see some of the most discriminating word; the word "said" was the most heavily weighted feature of the "true" class; which acts as a good sanity check that our model is indeed working correctly since we expect true article to make more direct quote. For the fake article class, "trump" was the most weighted word although it was also frequent in true articles. However, "president" was much more frequent in the fake news class, hence we may potentially extrapolate that the bigram "president trump" was more frequent in this class.

4.3 Random Forest

The input was vectorized using using a TFID vectorizer and splitted as 80/20 train-test sets respectively. Hyper-parameters search for the random forest classifier was done using randomized grid search and 5-fold cross-validation. Details about the hyper-parameters found can be found in Table 5. The optimized random-forest classifier correctly classified the test set with an *0.82 accuracy rate*. The training of the classifier took 9.31s on a CPU while the hyper-parameter search took about 25 minutes. This result is not significantly higher than our baseline, particularly considering we did not perform hyper-parameter optimization for our baseline.

4.4 XGBoost

The input was vectorized using using a TFID vectorizer and splitted as 80/20 train and test sets respectively. Hyper-parameters search for the XGBoost classifier was done using randomized grid search and 4-fold cross validation. Details about the hyperparameters can be found in Table 5 . It correctly classified the test set with an *0.83 accuracy rate*. We make very similar remarks about the XGBoost classfier as the random forest: it only marginally improves on the baseline despite optimization and training times that are more than an order of magnitude longer. This points to the fact that the TFID vectorization is the driving factor of the models' performances; they arrive at similar classification pattern based on the word frequency. Hence, in order to significantly improve on our simple baseline model, we must examine more complex language models and not only more complex machine learning classifiers.

4.5 Word2Vec BiLSTM

GenSim Word2Vec library was used to vectorize the input. We then use this vectorized representation as an embedding layer of equal dimension in our bi-directional LSTM model. A bi-LSTM neural network is constituted of LSTM units that incorporate past and future context as the units operate in both directions, hence it may retain longer-term context and improve text-classification [11]. Here, a standard dimension size of 100 was used. The model was then constituted of the embedding layer, a bi-LSTM layer with 128 memory units, a dropout layer with a dropout rate of 0.2 to reduce over-fitting and a final dense layer with a sigmoid activation function for the binary classification. Parameter choices not mentioned here can be found in 6. This model classified articles with an *accuracy rate of 0.85*. A more multilayered and complex model may obtain better results; however due to time and hardware constraints we limited our experiment to few layers and 6 epochs of training which took a few hours on Google Collaboratory.

4.6 ELECTRA

Masked Language Models (MLMs) such as transformers' BERT [12], RoBERTa and ALBERT are largely considered state-of-the-art in text-classification, reading comprehension and question answering. Their strengths lies in providing pre-trained bidirectional encoders that are not optimized in any task-specific way; hence only necessitate an extra output layer-the "fine-tuning" step- to provide particular state-of-the-art models. ELECTRA [13] builds on this architecture by optimizing both the pre-training and fine-tuning phase, requiring much less computing power and time. Note that the inclusion of this model in the project serves as a stepping-stone to further improvements and is not explored in its entirety. A wrapper library of transformers called simpletransformers was used to facilitate the fine-tuning of the ELECTRA model. Default parameters from huggingface documentation were used (some of the main ones are shown in Table 7. The data was splitted as 80/20 train/test. This model classified test articles with an *accuracy rate of 0.85*. While it may seem surprising this model did not perform better, it is important to reiterate that this model was used as a proof of concept of potential improvements; not as a demonstration of good optimization.

5 Conclusions

Our initial results with the baseline model and models using a TFID vectorization (random forest and XGBoost) point to two observation: such vectorization is reasonably efficient and might be useful in either low-stake high-volume classification or as a feature engineering or extraction tool for more complex models. Among models which used the same vectorization, results were relatively similar despite parameter optimization. Our bi-directional LSTM and ELECTRA model performed well, however these models were not optimized to their full potential due to hardware constraints and the ongoing learning process regarding these models. Overall, our experimentation leave a incomplete but positive picture for text-classification and more particularly fake-news classification: recent datasets significantly improve over their predecessors in terms of labelling and width of domain; simple models have reasonable performances and potential use in feature engineering, and state-of-the-art models are successfully applicable to the problem. As a future endeavour, using simple models to independently evaluate feature importance in the title and the body of an article to extract pertinent feature for a more complex model might be a worthy inquiry.

6 Figures & Tables

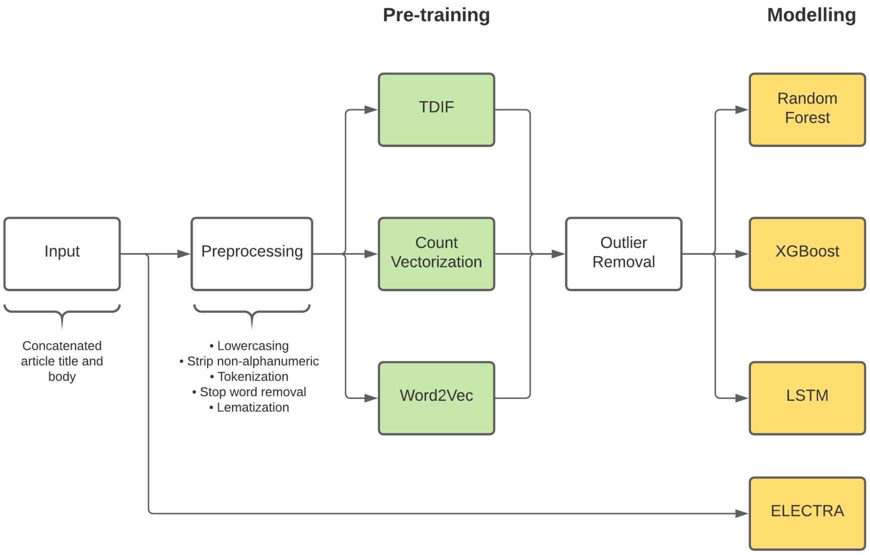

Figure 1: Modelling Flowchart

Table 3: NELA-2019 Dataset Instance

Column	Description
id	article id
date	publication date string in YYYY-MM-DD format
source	name of the source from which the article was collected
title	headline of the article
content	body text of the article
author	author of the article (if available)
published	publication date time string as provided by source (inconsistent formatting)
published_utc	publication time as unix time stamp
collection_utc	collection time as unix time stamp

Table 4: Word2Vec model word cosine similarity

trump	canada	vaccine	scotch
obama (0.666)	australia (0.659)	vaccination (0.826)	tart (0.892)
president (0.665)	canadian (0.628)	mmr (0.817)	caramel (0.890)
potus (0.660)	finland (0.593)	measles (0.729)	yoghurt (0.871)
realdonaldttrump (0.576)	denmark (0.581)	merck (0.712)	whisky (0.865)
democrat (0.572)	italy(0.580)	polio (0.699)	shallot (0.865)

Table 5: Non-default sklearn models hyper-parameters

Parameter	Random Forest	XGBoost
n_estimators	800	1000
min_samples_split	2	-
min_samples_leaf	2	-
max_features	sqrt	-
max_depth	95	4
subsample	-	0.8
learning_rate	-	0.1
gamma	-	1
colsample_bytree	-	0.8

Table 6: Hyper-parameters of bi-LSTM model

Parameter	Word2Vec
Tokenizer	Keras Tokenizer
Embedding Dimension	100
Encoder	1 layer bi-LSTM with 128 units
Epochs	6 epochs
Optimizer	Adam
Loss	Binary cross-entropy

Table 7: Hyper-parameters of ELECTRA

Parameter	ELECTRA
Tokenizer	ELECTRA tokenizer (similar to BertTokenizer)
Embedding Size	128
Hidden Layers	12
Epochs	3 epochs
Max Sequence Length	512

References

[1] Fabio Tagliabue, Luca Galassi, and Pierpaolo Mariani. The “pandemic” of disinformation in covid-19. *SN Comprehensive Clinical Medicine*, 2, 08 2020.

[2] Soroush Vosoughi, Deb Roy, and Sinan Aral. The spread of true and false news online. *Science*, 359:1146–1151, 03 2018.

[3] Kai Shu, Guoqing Zheng, Yichuan Li, Subhabrata Mukherjee, Ahmed Hassan Awadallah, Scott Ruston, and Huan Liu. Leveraging multi-source weak social supervision for early detection of fake news, 2020.

[4] Yi-Ju Lu and Cheng-Te Li. Gcan: Graph-aware co-attention networks for explainable fake news detection on social media. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[5] Álvaro Ibrain Rodríguez and Lara Lloret Iglesias. Fake news detection using deep learning, 2019.

[6] Thai Le, Suhang Wang, and Dongwon Lee. Malcom: Generating malicious comments to attack neural fake news detection models, 2020.

[7] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media. *Big Data*, 8(3):171–188, Jun 2020.

[8] William Yang Wang. “liar, liar pants on fire”: A new benchmark dataset for fake news detection. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2017.

[9] Maurício Gruppi, Benjamin D. Horne, and Sibel Adalı. Nela-gt-2019: A large multi-labelled news dataset for the study of misinformation in news articles, 2020.

[10] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. New Jersey, USA, 2003.

[11] Depeng Liang and Yongdong Zhang. Ac-blstm: asymmetric convolutional bidirectional lstm networks for text classification. *arXiv preprint arXiv:1611.01884*, 2016.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[13] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2019.