Identification of Propaganda-Based Fake News

Final Report

Sravani Boinepelli 20171050

Zubair Abid 20171076

Shelly Jain 20171008

Abhigyan Ghosh 20171089

{sravani.boinepelli, zubair.abid, shelly.jain, abhigyan.ghosh}
@research.iiit.ac.in

Abstract

Fake news and propaganda both pose considerable dangers to the average netizen. Today, detection of these has become paramount to ensure online security. This report describes our approach to the identification of propaganda-based fake news. We have implemented a few models and then integrated them into a browser extension. This can be used to automatically flag questionable articles on news websites and warn the reader.

Link to the datasets, code and slides, and demonstration video.

1 Introduction

The issue of fake news detection on social media is both challenging, and incredibly relevant today. Especially with its ability to affect people's behaviours in frightening, sometimes lethal ways, monitoring and tracking it down is of utmost importance. Likewise, another common issue with social media and the modern web is the manner in which it optimises for engagement means that propaganda is abound. The speed at which propaganda is distributed makes us particularly susceptible to the dangers of propaganda-based fake news, and this uncontrollable spread of false information can cause many complications at both macro and micro scales.

2 Literature Review

Since 2013 onwards, most NLP work involving classification has utilised word embeddings such as Word2Vec (Mikolov et al. 2013) and GLoVe (Pennington, Socher, and Manning 2014) as language features over which larger models are trained, making use of distributional semantics. While performant as demonstrated in Hardalov's work on credibility (Hardalov, Koychev, and Nakov 2016), these embeddings still lack per-word usage context. Efforts such as Seq2Seq (Sutskever, Vinyals, and Le 2014) and Attention over Seq2Seq (Bahdanau, Cho, and Bengio 2014) have made some contribution to this regard but the NLP landscape has, since 2017, been inundated by the Transformer (Vaswani et al. 2017) and related models such as BERT (Devlin et al. 2018).

With such a boom in social media, the spread of fake news is a growing threat. Due to this, numerous studies have arisen regarding its censuring. The following papers describe the methods that have been used to handle instances of fake news and propaganda related tasks, and the challenges posed.

"Fake News Detection on Social Media: A Data Mining Perspective" (Shu et al. 2017) is an overview of the how other scholarly works have tackled the problem of fake news detection. The survey presents a comprehensive review of detecting fake news on social media - including fake news characterisations on psychology and social theories, existing algorithms from a data mining perspective, evaluation metrics and representative datasets. Additionally, the authors also discuss related research areas, open problems, and future research directions. "Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking" (Rashkin et al. 2017) presents an analytic study characterizing the language of political quotes and news media written with varying intents and degrees of truth. Their best overall performing model architecture primarily composes of LSTM supplemented by LIWC features, using accuracy over PolitiFact. "A Benchmark Study on Machine Learning Methods for Fake News Detection" (Khan et al. 2019) compares the performances of three different models on three different datasets. Of these, the largest and most diversified dataset was created by the authors themselves. The comparison of the models was done on the basis of all metrics, but further analysis focused on accuracy. The best performing models were a character-level LSTM and a bi-LSTM, for distinct fake/real news corpora and combined corpus respectively.

Work on computational propaganda detection is relatively less common. "A Survey on Computational Propaganda Detection" (Giovanni Da San Martino et al. 2020) aims to review the state of the art on computational propaganda detection from the joint perspective of Network Analysis and NLP. The paper also discusses current challenges and future research directions. "Proppy: Organizing the news based on their propagandistic content" (Barrón-Cedeño et al. 2019) carries out multiple experiments on propaganda detection in news articles, and they conclusively show that representations modeling writing style and text complexity are more effective than word n-grams, which model topics. They further presented a system that organizes news articles into events and, for each event, shows articles according to their level of propagandistic content. The metric used for comparison were F1 score and accuracy.

We then look at some previous submissions to prior SemEval tasks that have focused on the task of propaganda detection. "Team QCRI-MIT at SemEval-2019 Task 4: Propaganda Analysis Meets Hyperpartisan News Detection" (Saleh et al. 2019) trained a logistic regression model with features ranging from simple bag of words to vocabulary richness, the kind of language it uses according to different lexicons, and its level of complexity and text readability. With proper pre-processing and scaling, they were able to achieve significant performance gains of up to 2% absolute in terms of accuracy. "SemEval-2020 Task 11: Detection of Propaganda Techniques in News Articles" (G. Da San Martino et al. 2020) presents the summary of all the results and findings of the SemEval-2020 Task 11 on Detection of Propaganda Techniques in News Articles. Both subtasks (Span Identification and Technique Classification) have been presented, along with analysis of the results, system submissions, and methods used by all the participating teams. The study shows the effectiveness of pretrained Transformers and ensembles for both tasks. "ApplicaAI at SemEval-2020 Task 11: On RoBERTa-CRF, Span

CLS and Whether Self-Training Helps Them" (Jurkiewicz et al. 2020) – which was awarded 'Best Paper' at SemEval 2020 on the Propaganda Shared Task – uses RoBERTa to develop two systems: one that identified propaganda in text, and one that classified spans to detect the fragments that one may suspect to represent one or more propaganda techniques. Although CRF is barely used with Transformer-based language models, the span identification task was approached with RoBERTa-CRF architecture while an ensemble of RoBERTa based models were used for the technique classification task. It provided the second best results on the F1 score on the development set for the shared task.

3 Datasets

The following are links to datasets which have been used for the purpose of the task.

Proppy Corpus 1.0 was the dataset used in the paper 'Proppy: Organizing the news based on their propagandistic content' (Barrón-Cedeño et al. 2019) which proposes models and experiments to assess the level of propagandistic content in an article. This dataset was used for the Semeval 2020's Task 11 on Propaganda detection.

The Liar dataset (Wang 2017) is a publicly available dataset containing 12,800 human-labelled sentences, with six levels of truthfulness ratings – *pants-fire*, *false*, *barely-true*, *half-true*, *mostly-true*, and *true*.

NELA-GT-2019 (Gruppi, Horne, and Adalı 2020) is a large multi-labelled news dataset, updated from the previous NELA-GT-2018 dataset. It contains 1.12M news articles from 260 sources (dated January 1st 2019 – December 31st 2019), from a wide range of mainstream and alternative news sources. Included along with the dataset are source-level ground truth labels from 7 different assessment sites covering multiple dimensions of veracity.

The MBFC Source List (Baly et al. 2018) corpus was created by retrieving websites along with their factuality and bias labels from the Media Bias/Fact Check (MBFC) website. The data contains the following fields:

1. source_url: the URL to each website (example: http://www.who.int/en/)
2. source_url_normalized: a shortened version of the source_url (example: who.int-en). These will be used as IDs to split the data into 5 folds of training and testing (in ./data/splits.txt)
3. ref: the link to the page in the MBFC website analysing the corresponding website (example:
4. http://mediabiasfactcheck.com/world-health-organization-who/)
5. fact: the factuality label of each website (low, mixed, or high)
6. bias: the bias label of each website (extreme-right, right, center-right, center, center-left, left, extreme-left)

4 Project Scope

This project aims to tackle the issue of automatically tagging text (in this case, from news article) as articles of propaganda-based fake news. For this purpose:

1. The **inputs** are the text from the news articles. These are extracted automatically from the web page of the news articles by the web browser extension. Other supplementary meta information from the articles are also extracted in order to improve classifier accuracy in later stages.

2. These inputs extracted by the interface are sent to a previously trained model, which automatically classifies the article in terms of both fake news vs real news, as well as level of propagandistic content.

3. The predicted classification from the model based on provided inputs are sent back to the interface. This is the system **output**, and it appears in the browser window for the user to see.

4.1 Issues Being Covered

The points of the problem statement which were tackled by the intended system are as follows:

1. The models used for the purpose of classification were built and trained ourselves. The training for these models was done on publicly available datasets like Proppy Corpus 1.0 (Barrón-Cedeño et al. 2019), Liar dataset (Wang 2017) and NELA-GT-2019 (Gruppi, Horne, and Adalı 2020). This was used to automatically classify the inputs into whether they might be propaganda-based fake news or not.

2. The available datasets all consist of articles from news websites which provide content pertaining to American political news – hence, this system too is used in the classification of the same, i.e. American political news.

3. The browser extension was built for use on the desktop version of the internet browser Mozilla Firefox. There is a functionality which allows the extension to be switched on and off whenever required.

4. The browser extension is able to automatically tag the content of articles from the web pages of reputed news sources.

4.2 Issues Not Being Covered

The points of the problem statement which were not tackled by the intended system are as follows:

1. The system is not multimodal. It only deals with identification of text articles, and is unable to classify other forms of media such as images, audio and video.

2. The model only performs a binary classification on whether the provided input article is propaganda-based fake news or not. It does not elaborate on the type of fake news and/or propaganda, etc. This is due to limitations on the available dataset and provided labels.

3. The system does not provide a span identification of the fake and/or propagandistic segment(s) of text.

4. The web browser extension can only extract the content from the web pages of the news articles. It cannot classify the content from the home pages of the news websites, or other non-news websites also proliferated with false content such as social media websites or advertisements.

5. Some news websites in our sources such as CNN do not adhere to the standard web format. The extension might not work for these websites. However most sites do follow the standard rules associated with web format which makes reading it easier.

6. The web browser extension is only compatible with a single web browser – Mozilla Firefox. It is not an extension which runs universally on other web browsers.

5 Implementation

The project was executed over 9 principal stages, as described in the following subsections.

5.1 Stage 1: Analysis and Preparation of the Datasets

First, we examined the two datasets of Proppy and Liar. The initial plan was for the final project to use a combination of Proppy and NELA as the latter includes PolitiFact ratings (which are the labels used in the Liar dataset), but as we found out later the intersection of the two did not provide sufficient quantity or quality for a deep learning model. For now, the intent was to figure out explicit patterns in the dataset (such as certain publications being classified as Propaganda altogether, which would mean we cannot use the publications as features for fear of overfitting).

Proppy includes with it MBFC labels and information about the sources of the articles in the dataset. This is not as useful for the analysis itself as it is annotated at the level of the source of publication of the article, and as such would act as a proxy for the publication. We find that the Proppy dataset is also annotated at the publication level, hence both these features are likely not of any use to us as they will cause the model to learn the "wrong" patterns; i.e. always tagging articles from any one publication as propaganda or otherwise even if that is not the case. Hence, for propaganda classification, we simply used the article text as features for the model.

For Liar, much of the information provided in the dataset (which is composed of single sentences as its primary text) is focused on metadata such as political inclination, speaker, and so on and so forth. As this metadata is not publicly available for most datasets, we are once

Source	Factuality Label	Bias Label
Breaking 911	Very Low	-
SHTFPlan	Mixed	Extreme Right
Clash Daily	Low	Extreme Right
Personal Liberty	Low	Extreme Right
Frontpage Magazine	Low	Extreme Right
The Washington Standard	-	-
VDare	Low	Extreme Right
Freedom Outpost	-	-
Remnant	Low	Extreme Right
Lew Rockwell	Mixed	Extreme Right

Table 1: All the News sources classified as ”propaganda”, and their corresponding MBFC information

again focusing entirely on the text itself.

For our final project submission, we had planned to use a combination of Proppy and NELA, as it provides us with both propaganda and fake news information. The NELA dataset contains several extra tags pertaining to the reliability of the news source which might have been useful for the final task, such as factuality ratings from several news verification agencies as well as the MBFC label. One set of factuality labels are from PolitiFact, which is the source used to derive the labels for the Liar dataset; this makes the information conveyed by Liar a subset of the information conveyed by NELA. Additionally, since this MBFC label is common to both NELA and Proppy, we are able to extract all those records whose source exists in both datasets. In this way, NELA is able to combine information from both Liar and Proppy, as well as include additional informational which will assist in the classification task. See Figure 1.

However, there were issues with doing an inner SQL join on both the datasets. The formatting of both the datasets were not compatible with each other. Once we sorted out the formatting issues, we found that we had to discard all the articles that did not have an entry for their source in NELA. This reduced the number of actually usable articles to 4000 which is a lot less compared to the 19.6K we had finally left. As a result, we could not end up using this combination of the datasets for our final models.

Link to the datasets

5.2 Stage 2: Sanity Check on Feasibility of the Task

As a sanity check, we ran a few baseline models on the Proppy and Liar datasets, using a range of more traditional machine learning classifiers, before we proceeded with any further training. For this, we have used `sentence-transformers`' (Reimers and Gurevych 2019) `distilbert-base-nli-stsb-mean-tokens` as the input embedding, after which we run it through a variety of popular classifiers, namely:

1. Multi-Layer Perceptron (MLP)

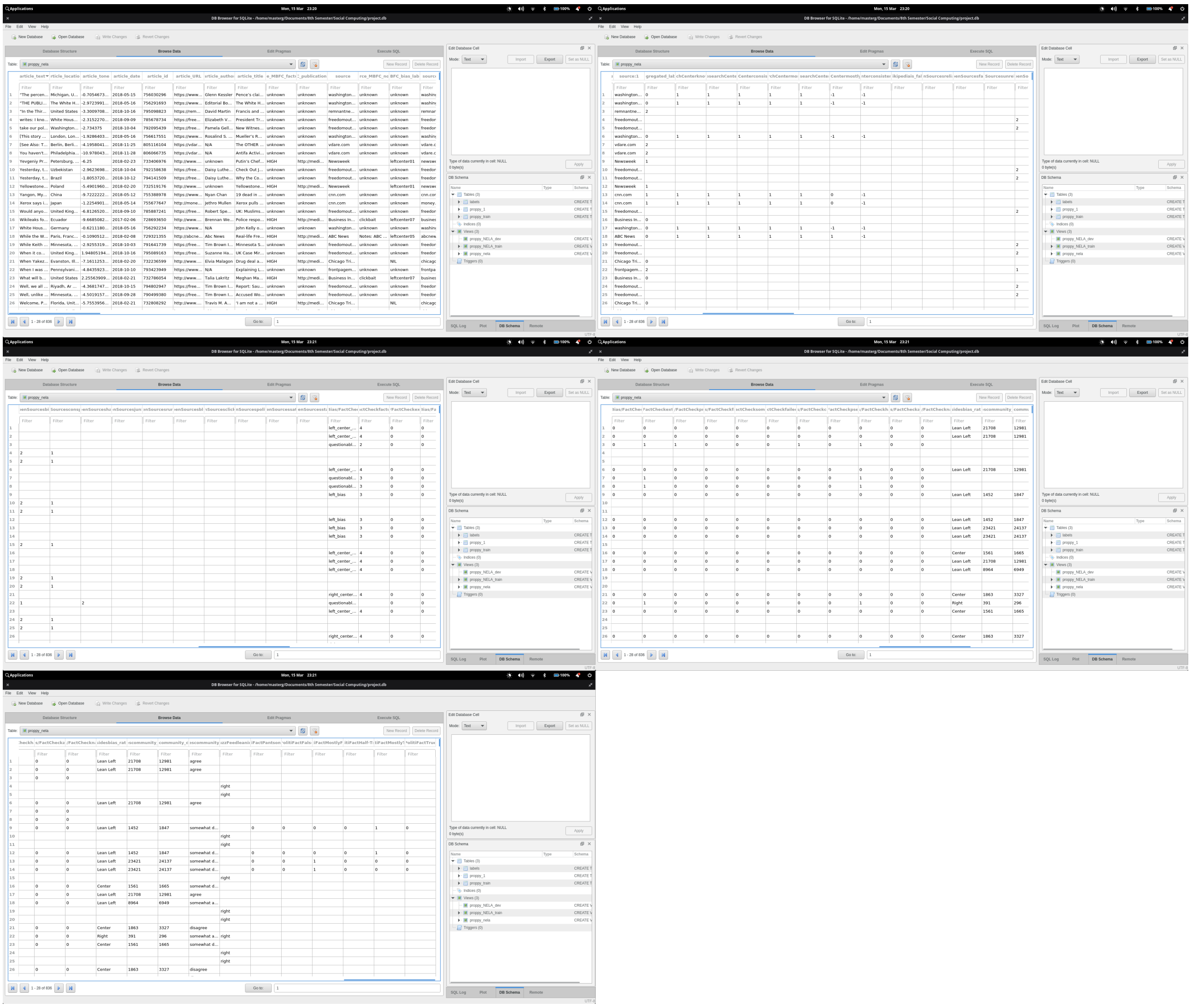


Figure 1: What the tables would have been after combining

2. Random Forest (RF)
3. Support Vector Machine (SVM)

For the classifiers we use `scikit-learn`'s (Pedregosa et al. 2011) provided modules. With each of these we run the model once without finetuning, and then optimise if the performance is upto par. Based on a grid-search on the MLP model, we found that the performance gain is not significant enough, given the initial results are not too promising. Because of such and as it is the baseline, we have opted against finetuning on these models.

5.2.1 Proppy

With MLP, initial results with sklearn's defaults and `max-iter` set to 100 gave us a 0.96 (macro average) $F1$ score on detecting non-propaganda news pieces, but only 0.64 for propaganda detection. A grid search over various parameters (provided below) did not yield significantly better results. The SVM classifier yielded similar results. Random Forest, however, performed

Classifier	Proppy			Liar		
	Propaganda	Not Propaganda	Overall	Fake	Not Fake	Overall
MLP	0.64	0.96	0.80	0.58	0.64	0.61
SVM	0.60	0.96	0.78	0.57	0.69	0.63
RF	0.15	0.94	0.54	0.50	0.66	0.58

Table 2: Macro F1 scores for the basic classifiers

significantly worse: identifying 0 propaganda articles with a max-depth of 2, 1 with a max-depth of 5, and a macro $F1$ score of 0.15 with max-depth set to None. Detailed information has been given in table 2.

```
parameters = {
    ‘activation’: [‘identity’, ‘tanh’, ‘relu’],
    ‘solver’: [‘lbfgs’, ‘adam’],
    ‘max_iter’: [100, 200, 500, 1000],
    ‘hidden_layer_sizes’: [(100,), (50, 100, 50), (100, 50, 50, 100)],
}
```

Link to code

5.2.2 Liar

For Liar we focused on a coarse-grained classification, reducing the six degrees of labels to two: ‘*true*’ or ‘*false*’. So ‘*pants-fire*’, ‘*false*’, and ‘*barely-true*’ are all classified as ‘*false*’, and the remaining three (‘*half-true*’, ‘*mostly-true*’, and ‘*true*’) are all ‘*true*’.

Performance is equally poor on both labels this time, in the range of 0.55 – 0.69. The RF Classifier is still the worst performing, albeit not by as much. Check table 2.

Link to code

5.3 Stage 3: Finetuning and Benchmarking the Models

As relying solely on pretrained models did not give good results, our next two experiments included training and then finetuning our fake news and propaganda datasets to check their independent performances from a binary classification problem standpoint. In the first experiment, we initialised a pretrained model from checkpoint weights and trained for 5, and then 8 further epochs on our datasets with an initial set of chosen hyperparameters, and in the next we finetuned the hyper-parameters and trained for 3 epochs instead.

5.3.1 Simple Transformers

The first relies on the simpletransformers [1] library, leveraging the BERT pretrained model. We used the AdamW optimizer (which decouples the weight decay from the optimization step),

[1] https://github.com/ThilinaRajapakse/simpletransformers

Data	**Macro F1 Score**
Proppy	0.8879
Liar	0.6158

Table 3: Macro F1 scores for SimpleClassifiers

Metric	**Proppy**			**Liar**		
	Propaganda	Not Propaganda	Overall	Fake	Not Fake	Overall
Macro-F1 Score	0.91	0.99	0.95	0.58	0.70	0.64

Table 4: Macro F1 scores for the finetuned classifiers

while the maximum sequence length used was 128 with a batch size of 3 epochs, learning rate of 4e-5 and a batch size of 8. Results were more promising for Proppy, but not Liar (Table 3).

Link to code for Proppy
Link to code for Liar

5.3.2 Finetune

For the latter experiment, we finetuned the pretrained model on our datasets using Indico's finetune library [2]. Using BERT-Small as the pretrained model we trained for 6 hours on 3 Nvidia 2080 Tis. There was a large improvement in the propaganda detection results (Table 4) as it iteratively searches for optimal parameter values. The model was then saved and used as the backend to the browser extension architecture.

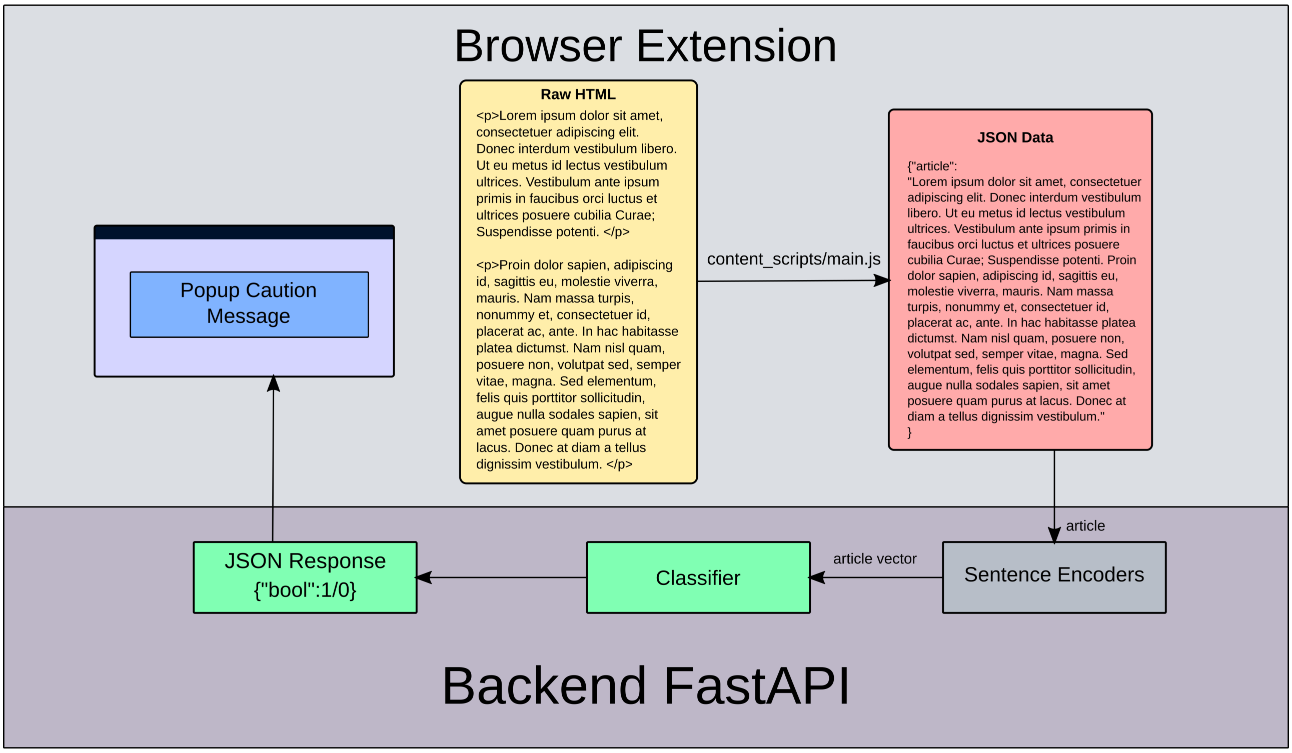
Link to the code(s)

5.4 Stage 4: Designing a Basic Browser Extension

The project description required us to deploy the model trained as a browser extension so it is accessible to more people than just the technically minded. For the purposes of this project, we have chosen to use the Mozilla Firefox platform due to its extensive documentation and community support. Additionally, since Firefox is an open-source project, it provides much more support for the developer.

The extension itself at this point was essentially a page scraper followed by an inference call on the saved model. The architecture is as follows (see Fig. 3) – when enabled for a page (it is compatible with most news websites, albeit website selection has not been setup yet), it will parse through all `<p>` tags, compile it into a coherent article, and send it as JSON to the backend which runs the inference.

[2] https://github.com/IndicoDataSolutions/finetune



```
<p>Lorem ipsum dolor sit amet,
consectetuer adipiscing elit.
Donec interdum vestibulum libero.
Ut eu metus id lectus vestibulum
ultrices. Vestibulum ante ipsum
primis in faucibus orci luctus et
ultrices posuere cubilia Curae;
Suspendisse potenti. </p>

<p>Proin dolor sapien, adipiscing
id, sagittis eu, molestie viverra,
mauris. Nam massa turpis,
nonummy et, consectetuer id,
placerat ac, ante. In hac habitasse
platea dictumst. Nam nisl quam,
posuere non, volutpat sed, semper
vitae, magna. Sed elementum,
felis quis porttitor sollicitudin,
augue nulla sodales sapien, sit
amet posuere quam purus at
lacus. Donec at diam a tellus
dignissim vestibulum. </p>
```

```
{"article":
"Lorem ipsum dolor sit amet, consectetuer
adipiscing elit. Donec interdum vestibulum
libero. Ut eu metus id lectus vestibulum
ultrices. Vestibulum ante ipsum primis in
faucibus orci luctus et ultrices posuere
cubilia Curae; Suspendisse potenti. Proin
dolor sapien, adipiscing id, sagittis eu,
molestie viverra, mauris. Nam massa
turpis, nonummy et, consectetuer id,
placerat ac, ante. In hac habitasse platea
dictumst. Nam nisl quam, posuere non,
volutpat sed, semper vitae, magna. Sed
elementum, felis quis porttitor sollicitudin,
augue nulla sodales sapien, sit amet
posuere quam purus at lacus. Donec at
diam a tellus dignissim vestibulum."
}
```

Figure 2: Older architecture of the browser extension

5.5 Stage 5: Implementing a Working Demo with the Baseline

We now had the basic framework for the browser extension implemented. However, since the models were not fully prepared, the backend for the web API was linked only to a single trained model. For the first submission, we linked the API backend to the two trained classifier models which together formed the baseline model for the task of identifying propaganda-based fake news.

The baseline model requires the article text to be sent for prediction to both models. Only then, once the predicted labels from both classifiers had been obtained, can the final output be determined (the method to determine final label was simple set intersection, at this stage). Unfortunately, both of the models being used were bulky, and the time taken to receive the response from the API was upward of two minutes. This is extremely poor performance, as this duration would be enough for the article to be consumed by the reader without them even being warned about the possible dangers of its content.

Clearly, at this point we had concluded that a solution must be devised in order to reduce the return time. For this, we approached our project mentors and received their advice on the matter. It was suggested that we move our framework away from Flask and towards an alternative that is more streamlined for work which involves a large server load of requests and responses – python's FastAPI.

	Propaganda	**Not Propaganda**
Fake	83	533
Not Fake	45	623

Table 5: Contingency table of Fake news vs Propaganda. Fake news is the ground truth, whereas Propaganda labels are predicted by finetuned models from stage 4.

5.6 Stage 6: Correlation Analysis

Since the aim of this project is to identify not just propaganda or just fake news, but *propaganda-based fake news*, we have implemented a rudimentary measure which is used to explore this. Intuitively, one expects for propaganda to be fake news. This suggestion is investigated using the previously trained baseline models for the separate classification of article text as propaganda and as fake news.

There are a few unavoidable limitations to this method of determining the correlations. Firstly, the veracity of the datasets themselves is questionable. The document detailing our analysis of the datasets notes that the Proppy dataset determined the article level ground truth classification using publisher level judgements; hence, any labels of 'propaganda' are unreliable at best. Additionally, the Liar dataset has samples which consist of only few lines of text, which makes it less effective when used to classify entire articles. Furthermore, the models used for the correlations analysis are trained on these unreliable datasets. The macro-F1 score obtained for Proppy was 95%, but it was only 64% for Liar. Unfortunately, these datasets must form the basis of our analysis due a lack of available alternatives.

Link to notebook with correlation analysis

Model one, which was trained on the propaganda dataset (here, Proppy), was tested on the fake news dataset (here, Liar). Once the predicted labels ('propaganda' or 'not propaganda') had been obtained for the data (originally tagged as 'fake' or 'not fake'), the co-occurrence of 'fake' and 'propaganda' labels was checked in the Liar dataset. This gives a good estimate for the degree of entailment from an articles measure of fakeness to its measure of propaganda, i.e. to what extent 'fake' implies 'propaganda' in news articles. The contingency table can be seen at Table 5.

Similarly, model two (trained on Liar) was tested on Proppy. Once again, the co-occurrence of 'fake' and 'propaganda' labels was checked in the Proppy dataset. This was used to check the entailment between propaganda and fake news in the reverse direction, i.e. to what extent 'propaganda' implies 'fake' in news articles. The contingency table can be seen at Table 6.

In order to make more sense of the data, we look at the percentages of:

- Fake (and True) news that was classified as propaganda
- Propaganda (and non-Propaganda) news that was classified as fake

This has been included in Table 7. As we can see from the data, while it is twice as likely for fake news to be marked as propaganda as opposed to true news, the bulk of fake news

	Fake	**Not Fake**
Propaganda	99	476
Not Propaganda	540	4010

Table 6: Contingency table of Propaganda vs Fake news. Propaganda labels are the ground truth, whereas Fake news labels are predicted by finetuned models from stage 4.

	Percentage
Fake news classified as propaganda	13.47%
True news classified as propaganda	6.74%
Propaganda classified as Fake news	17.22%
Not Propaganda classified as Fake news	11.97%

Table 7: Percentages to determine correlation of Fake news and Propaganda

– about 86.5% of it – continues to be non-propaganda pieces. This is due to the fact that most samples in the dataset are not tagged as propaganda (89% of the Liar eval set is not propaganda). Likewise, propaganda is more likely to be marked as fake news than non-propaganda.

We also do a χ^2 test to check for dependence, using libraries provided by SciPy. The p-value for both directions is significantly less than 0.05, and so we can verify that the variables are not independent of one another.

From the aforementioned tables, it can be observed that the degree of fakeness and the degree of propaganda of a news article are strongly correlated. Unfortunately, while there is visible correlation, there is no particular direction of entailment between the two factors. Thus, entailment between the two might not improve the model performance, as expected at the hypothesis stage of the project. The experiments conducted in Stages 7 and 9 appear to corroborate with this newer hypothesis.

5.7 Stage 7: Joint Training of the Proposed Model - Pretraining from Scratch

As an aim of the project was to use propaganda information to infer fake news, and due to the previously seen (and prior assumed) correlation of fake news and propaganda, it is possible that by using this shared information we could improve on our previously obtained results. In this stage, we tried to see if this was the case, or if it was not as useful for final real-world deployment.

Based on feedback from meetings with our mentors, we decided to finetune the fake news model in order to load custom weights obtained by training from scratch on the text from the propaganda dataset, instead of using pretrained weights from HuggingFace. Generally, we started training by randomly initialising weights, and soon after this the weights were changed in order to perform the task with less mistakes (i.e. optimisation). Once we were satisfied with the training results we saved the weights of the network. However, because we wanted to train the network to perform a different task, i.e. propaganda based fake news detection as opposed

Model	**Params**		**Results (in %)**			
	Epochs	Weight Decay	Acc	Macro F1	Precision	Recall
Prop-Liar	3	default	60.0	63.2	60.8	65.4
	3	0.01	62.0	59.2	61.0	57.4
Liar	3	default	63.0	56.2	66.0	48.0
	5	default	62.4	58.7	62.1	55.6

Table 8: Performance of pretrained Propaganda based Fake news model vs Simple Fake news model over various parameters

to just propaganda/just fake news detection, instead we used the weights we saved from the previous network as the initial weight values for the new model. Therefore, by pretraining on the propaganda dataset and then using this to train on the fake news dataset, we were able to get a model that was basically a solution to the propaganda based fake news problem despite lacking a primary dataset that was annotated for that specific task.

We began by training a byte-level BPE i.e. byte-pair encoding tokeniser (the same as GPT-2), with the same special tokens as RoBERTa, rather than using a WordPiece tokeniser like BERT. This was because it would start building its vocabulary from an alphabet of single bytes, so all words would be decomposable into tokens (no more `<unk>` tokens).

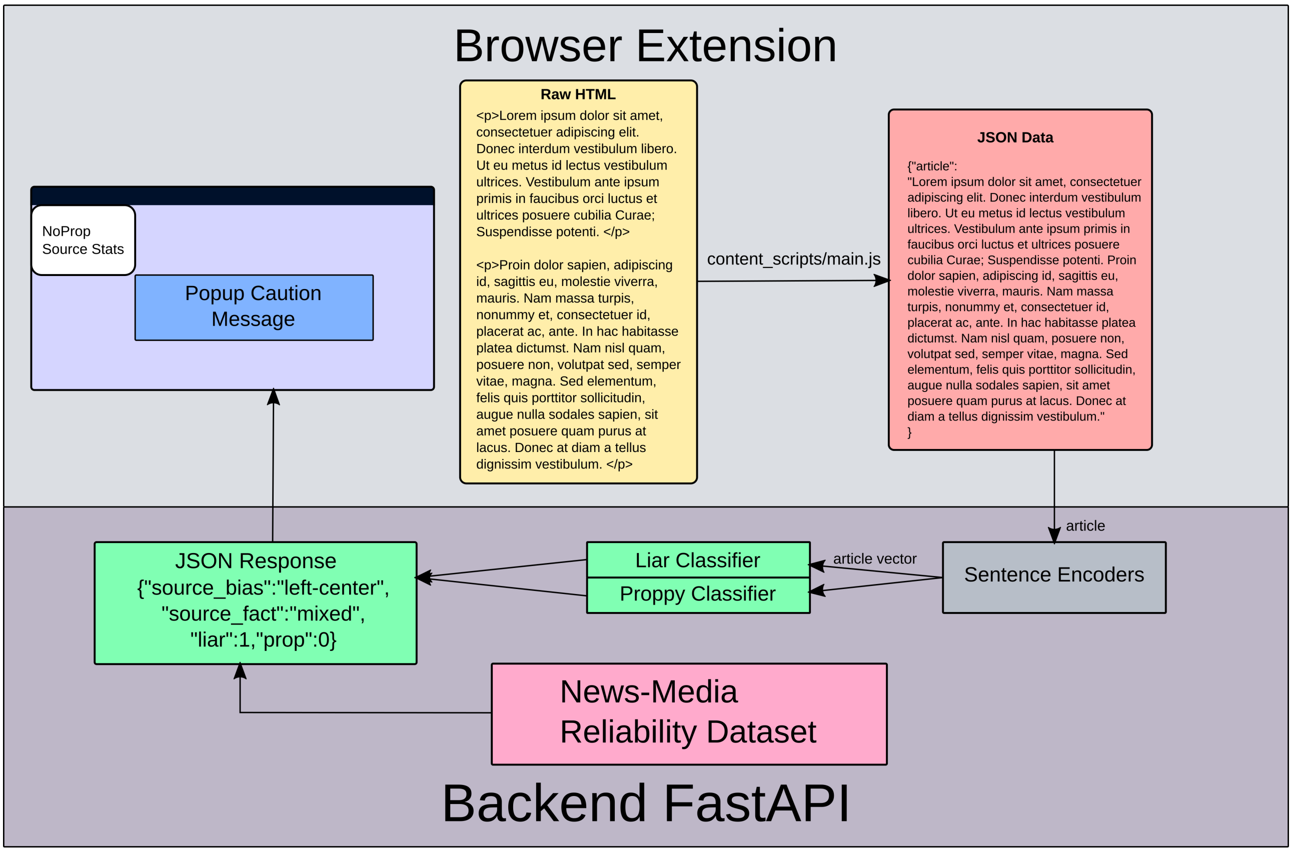
As we were training a language model from scratch in order to get our pretrained model, we only initialised the model from a config and not from an existing pretrained model or checkpoint. We then initialised our trainer with various parameters that we ran and tested on several experiments such as varying the strength of weight decay, batch size, epochs etc. In Table 8, we can see the results of these experiments compared to that of the simple fake news model where Prop-Liar represents the pretrained propaganda based fake news model and Liar represents the fake news model (not including pretrained propaganda model).

The above results were run with a device batch size 8 with 4 GPUs (so a total of 32) which was better than batch size 64. We noticed that keeping the weight decay at the default '0' rather than '0.01' gave us better F1 scores. Decreasing the batch size and increasing the number of epochs also gave us a better performance.

5.8 Stage 8: Reducing API Return Times

As mentioned earlier, the previous API implementation ran extremely slowly and had a high return time. It ran on python Flask, which caused a great amount of load on the server, since it Flask synchronous in design.

During a discussion with our mentor, he suggested the use of another python framework for web based API development: FastAPI. Unlike Flask, this framework is asynchronous. This means that before, the Flask code had to executed in sequence, blocking off all other independent instructions until the execution of the previous statement was completed; the new FastAPI code, being asynchronous, was able to exchange requests and responses independent of time



```
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec interdum vestibulum libero. Ut eu metus id lectus vestibulum ultrices. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Suspendisse potenti. </p>

<p>Proin dolor sapien, adipiscing id, sagittis eu, molestie viverra, mauris. Nam massa turpis, nonummy et, consectetuer id, placerat ac, ante. In hac habitasse platea dictumst. Nam nisl quam, posuere non, volutpat sed, semper vitae, magna. Sed elementum, felis quis porttitor sollicitudin, augue nulla sodales sapien, sit amet posuere quam purus at lacus. Donec at diam a tellus dignissim vestibulum. </p>
```

```
{"article":
"Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Donec interdum vestibulum libero. Ut eu metus id lectus vestibulum ultrices. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Suspendisse potenti. Proin dolor sapien, adipiscing id, sagittis eu, molestie viverra, mauris. Nam massa turpis, nonummy et, consectetuer id, placerat ac, ante. In hac habitasse platea dictumst. Nam nisl quam, posuere non, volutpat sed, semper vitae, magna. Sed elementum, felis quis porttitor sollicitudin, augue nulla sodales sapien, sit amet posuere quam purus at lacus. Donec at diam a tellus dignissim vestibulum."
}
```

```
{"source_bias":"left-center",
"source_fact":"mixed",
"liar":1,"prop":0}
```

Figure 3: Updated Architecture of the browser extension

or sequence, and completed the execution by firing an event or by calling a provided callback function.

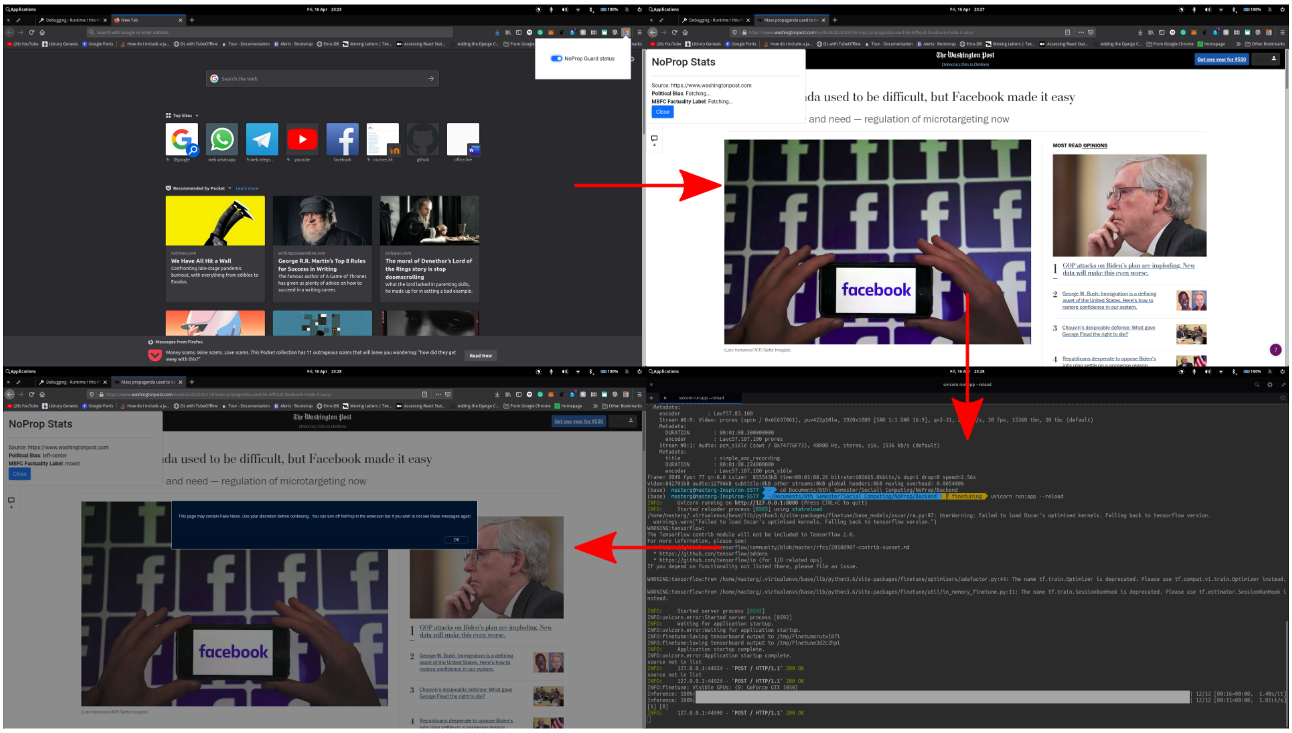
The replacement of Flask with FastAPI proved incredibly effective. Moving from synchronous to asynchronous execution brought down the return time from over two minutes to just approximately thirty seconds.

The architecture is as follows (see Fig. 3) A working demo is shown in (see Fig. 4)
Link to demonstration of browser extension

5.9 Stage 9: Streamlining and Polishing

At this stage, the basic implementation was prepared, so the focus was on improvement of the same. The first priority was to improve the performance of the classifier itself. Due to the time taken to pretrain on Proppy and then finetune on Liar, we had only been able to run some experiments with a limited set of hyperparameters at end of the previous stage, so we focused on running multiple experiments in order to ensure what we had was optimal.

The other side of the problem was improving the response of the browser extension. This could be broadly categorised into two focuses – bettering the interface, and speeding up the response time. These two edits collectively increased the quality of experience of the browser extension.



NoProp Stats

Source: https://www.washingtonpost.com
Political Bias: Fetching...
MBFC Factuality Label: Fetching...
Close

NoProp Stats

Source: https://www.washingtonpost.com
Political Bias: left-center
MBFC Factuality Label: mixed
Close

This page may contain Fake News. Use your discretion before continuing.. You can turn off NoProp in the extension bar if you wish to not see these messages again
OK

```
  Metadata:
    encoder         : Lavf57.83.100
    Stream #0:0: Video: prores (apcn / 0x6E637061), yuv422p10le, 1920x1080 [SAR 1:1 DAR 16:9], q=2-31,     /s, 30 fps, 15360 tbn, 30 tbc (default)
    Metadata:
      DURATION        : 00:01:08.300000000
      encoder         : Lavc57.107.100 prores
    Stream #0:1: Audio: pcm_s16le (sowt / 0x74776F73), 48000 Hz, stereo, s16, 1536 kb/s (default)
    Metadata:
      title           : simple_aac_recording
      DURATION        : 00:01:08.224000000
      encoder         : Lavc57.107.100 pcm_s16le
frame= 2049 fps= 77 q=-0.0 Lsize=  855543kB time=00:01:08.26 bitrate=102665.0kbits/s dup=1 drop=0 speed=2.56x
video:842701kB audio:12796kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.005400%
(base)  masterg@masterg-Inspiron-5577  ~  cd Documents/8th\ Semester/Social\ Computing/NoProp/Backend
(base)  masterg@masterg-Inspiron-5577  ~/Documents/8th Semester/Social Computing/NoProp/Backend  finetuning  uvicorn run:app --reload
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [8583] using statreload
/home/masterg/.virtualenvs/base/lib/python3.6/site-packages/finetune/base_models/oscar/ra.py:87: UserWarning: Failed to load Oscar's optimised kernels. Falling back to tensorflow version.
  warnings.warn("Failed to load Oscar's optimised kernels. Falling back to tensorflow version.")
WARNING:tensorflow:
The TensorFlow contrib module will not be included in TensorFlow 2.0.
For more information, please see:
  * https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-sunset.md
  * https://github.com/tensorflow/addons
  * https://github.com/tensorflow/io (for I/O related ops)
If you depend on functionality not listed there, please file an issue.

WARNING:tensorflow:From /home/masterg/.virtualenvs/base/lib/python3.6/site-packages/finetune/optimizers/adafactor.py:44: The name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /home/masterg/.virtualenvs/base/lib/python3.6/site-packages/finetune/util/in_memory_finetune.py:11: The name tf.train.SessionRunHook is deprecated. Please use tf.estimator.SessionRunHook i
nstead.

INFO:     Started server process [8592]
INFO:uvicorn.error:Started server process [8592]
INFO:     Waiting for application startup.
INFO:uvicorn.error:Waiting for application startup.
INFO:finetune:Saving tensorboard output to /tmp/Finetunerutxl87i
INFO:finetune:Saving tensorboard output to /tmp/Finetune3d2c2hpl
INFO:     Application startup complete.
INFO:uvicorn.error:Application startup complete.
source not in list
INFO:     127.0.0.1:44924 - "POST / HTTP/1.1" 200 OK
source not in list
INFO:     127.0.0.1:44924 - "POST / HTTP/1.1" 200 OK
INFO:finetune: Visible GPUs: {0: GeForce GTX 1050}
Inference: 100%|████████████████████████████████████████| 12/12 [00:16<00:00,  1.40s/it]
Inference: 100%|████████████████████████████████████████| 12/12 [00:11<00:00,  1.01it/s]
[1] [0]
INFO:     127.0.0.1:44990 - "POST / HTTP/1.1" 200 OK
```

Figure 4: Demo of the browser extension

To make the interface better, the additional meta data from the datasets was leveraged. This was used in order to provide additional information to the user about the political biases of the article, along with the degree of factuality conveyed by it. Such information allows the user to employ their own discretion and make sound judgements about the article to support the prediction by the classifier. Hence, the quality of the result was improved. Besides this, the back-end code for the extension was further optimised to increase the efficiency of execution. This allowed us to bring down the inference time from 30s to 10s by reducing the number of times the assests had to be loaded. Together, these two updates provided for a tool that was further polished version compared to its predecessor.

6 Challenges

We faced a few roadblocks along the way. This impeded our progress at each stage. Here, we list certain limitations involving the dataset and other resources which gave rise to some of these problems.

1. A major issue in ensuring the integrity of the model was the lack of soundness of the labels in the Proppy dataset. Though there are several separate labels which describe the level of bias and the level of factuality separately, both of these seem to be highly dependent on the MBFC label. This means that these are not unique labels but effectively just derivations of the previously mentioned MBFC label. This is problematic because the MBFC labels are only source level information - hence all three features convey information which adversely affects the training and we are consequently unable to use any of the three features. A further drawback is that in Proppy, the MBFC labels themselves are now

Source	Factuality Label	Bias Label
Breaking 911	Very Low	-
SHTFPlan	Mixed	Extreme Right
Clash Daily	Low	Extreme Right
Personal Liberty	Low	Extreme Right
Frontpage Magazine	Low	Extreme Right
The Washington Standard	-	-
VDare	Low	Extreme Right
Freedom Outpost	-	-
Remnant	Low	Extreme Right
Lew Rockwell	Mixed	Extreme Right

Table 9: MBFC Labels for all the Sources tagged "Propaganda"

outdated by 2 years.

2. For some news sources in the Proppy dataset, the entries appeared twice. An example is the existence of both 'cnn.com' and 'CNN' for the news channel CNN. All such repetitions had to be manually combined before the Proppy and NELA datasets could be crossed.

3. The Liar dataset consists of only single-sentence data. Since the classification is being done on entire articles of input, this is not an optimal modelling of data for training and is unable to reflect the expected input for prediction in the final model. Unfortunately, there is a severe dearth of available datasets for fake news prediction. Additionally, this dataset needs to share a common domain with the dataset for propaganda detection. Hence, the model is limited to training on single sentences.

   Some examples of misclassification (and one case of correct classification) in the Liar dataset are enumerated below:

   (a) '*Says nearly 30,000 federal employees in Oregon were furloughed during the government shutdown.*' (Fake) Tagged as (True)

   (b) '*Just like Donald Trump, David Jolly wants to outlaw a womans right to choose.*' (Fake) Tagged as (True)

   (c) '*Ronald Reagan did amnesty*' (True) Tagged as (Fake)

   (d) '*Congress will begin its recess without having allocated one penny to fight Zika*' (True) Tagged as (Fake)

   (e) '*Says Tennessee has some of the lowest beer excise tax rates in the country.*' (True) Tagged as (True)

4. The downloaded dataset NELA was poorly formatted. It did not follow the appropriate template of tsv files, and as a result loading the dataset proved to be quite problematic. We were forced to manually edit the tables so that all the samples could be read in the proper format. In addition to this, some of the cell values were also unreadable, and were producing 'nan' output in the python code.

5. There were data formatting issues that were not picked up by pandas dataframes but came up when using the library for pretraining. These proved to be difficult to overcome unless going in to manually examine the data and then removing it using regex pattern matching.

6. Some of the dependencies use older versions of Tensorflow. So we had to downgrade all the packages to those past versions. This makes the architecture very fragile and susceptible to breaking should one of the modules remove support.

7. The problems related to the models and classifications also seep into the browser extension. The response times are related to the inference time for classification which is high as BERT takes time for sentence encoding. Another problem is that using BERT for a web-app requires high computing power which can be expensive. But since our project is focused more on models, we ignore the hardware constraints.

7 Future Scope

Like any project, there is potential in the project as well to be extended. Possible avenues of extension include the following:

1. Further improving the classification models to improve the accuracy of prediction.

2. Identifying the spans of propagandistic content in the articles and higlighting them for the reader as regions of caution.

3. Determining the quality of the text – the degree of veracity of the news article.

4. Identifying the categories of propagandistic fake news which were detected in the news article. Some examples of these would be name calling, glittering generalities, card stacking, etc.

5. Verifying the factual content present in the news article being observed. This would be done against pre-existing (and possibly, regularly updated) knowledge bases or other similar alternative representation.

6. Extending the model to work in other domains, beyond just American political news. This could range from Indian political news all the way to celebrity gossip.

References

Ahmed, Hadeer, Issa Traore, and Sherif Saad (2017). “Detection of online fake news using n-gram analysis and machine learning techniques”. In: *International conference on intelligent, secure, and dependable systems in distributed and cloud environments.* Springer, pp. 127–138.

— (2018). “Detecting opinion spams and fake news using text classification”. In: *Security and Privacy* 1.1, e9.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2014). “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473.*

Baly, Ramy et al. (2018). “Predicting Factuality of Reporting and Bias of News Media Sources”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP ’18. Brussels, Belgium.

Barrón-Cedeño, Alberto et al. (2019). “Proppy: Organizing the news based on their propagandistic content”. In: *Information Processing Management* 56.5, pp. 1849–1864. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2019.03.005. URL: http://www.sciencedirect.com/science/article/pii/S0306457318306058.

Devlin, Jacob et al. (2018). “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *CoRR* abs/1810.04805. arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

Gruppi, Maurício, Benjamin D. Horne, and Sibel Adalı (2020). *NELA-GT-2019: A Large Multi-Labelled News Dataset for The Study of Misinformation in News Articles*. arXiv: 2003.08444 [cs.CY].

— (2021). *NELA-GT-2020: A Large Multi-Labelled News Dataset for The Study of Misinformation in News Articles*. arXiv: 2102.04567 [cs.CY].

Hardalov, Momchil, Ivan Koychev, and Preslav Nakov (2016). “In search of credible news”. In: *International conference on Artificial intelligence: methodology, systems, and applications*. Springer, pp. 172–180.

Jurkiewicz, Dawid et al. (Dec. 2020). “ApplicaAI at SemEval-2020 Task 11: On RoBERTa-CRF, Span CLS and Whether Self-Training Helps Them”. In: *Proceedings of the Fourteenth Workshop on Semantic Evaluation*. Barcelona (online): International Committee for Computational Linguistics, pp. 1415–1424. URL: https://www.aclweb.org/anthology/2020.semeval-1.187.

Khan, Junaed Younus et al. (2019). *A Benchmark Study on Machine Learning Methods for Fake News Detection*. arXiv: 1905.04749 [cs.CL].

Martino, G. Da San et al. (2020). *SemEval-2020 Task 11: Detection of Propaganda Techniques in News Articles*. arXiv: 2009.02696 [cs.CL].

Martino, Giovanni Da San et al. (2020). “A Survey on Computational Propaganda Detection”. In: *arXiv preprint arXiv:2007.08024*.

Mikolov, Tomas et al. (2013). “Distributed representations of words and phrases and their compositionality”. In: *arXiv preprint arXiv:1310.4546*.

Pedregosa, F. et al. (2011). “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Rashkin, Hannah et al. (Sept. 2017). “Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 2931–2937. DOI: 10.18653/v1/D17-1317. URL: https://www.aclweb.org/anthology/D17-1317.

Reimers, Nils and Iryna Gurevych (Nov. 2019). “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. URL: https://arxiv.org/abs/1908.10084.

Saleh, Abdelrhman et al. (2019). “Team QCRI-MIT at SemEval-2019 Task 4: Propaganda Analysis Meets Hyperpartisan News Detection”. In: *arXiv preprint arXiv:1904.03513*.

Shu, Kai et al. (2017). *Fake News Detection on Social Media: A Data Mining Perspective*. arXiv: 1708.01967 [cs.SI].

Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le (2014). “Sequence to Sequence Learning with Neural Networks”. In: *CoRR* abs/1409.3215. arXiv: 1409.3215. URL: http://arxiv.org/abs/1409.3215.

Vaswani, Ashish et al. (2017). “Attention is all you need”. In: *arXiv preprint arXiv:1706.03762*.

Wang, William Yang (2017). *"Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection*. arXiv: 1705.00648 [cs.CL].