



**UNIVERSITY
OF LONDON**

CM3070 Final Project
Final Project Report
Project Idea Title: "Fake News Detection"
A Chrome-based Web Extension

Lee Jun Cheng Alston
Student ID: 210201392

Abstract

This report aims to present a real-world problem of Fake News and its associated relevance in today's world. The objective is to design and develop a solution based on Project Idea Title 1: Fake News Detection from the CM3060 Natural Language Processing Module. This solution involved developing a user-friendly web extension that can verify the validity of the information of news articles online for users to ensure that the information being consumed is verified and safe for perusal.

This project begins with an overview of the domain of Fake News, followed by a summary of its dangers and impact on society based on its relevance today. The topic of comparisons to other pre-existing solutions will also be discussed, determining the features that existing solutions lack to tackle this prevailing issue. The solution that is to be implemented and developed in this project aims to tackle the lapses identified in the existing solutions today, providing a comprehensive and pragmatic solution to the pressing issue of Fake News.

By highlighting the focus on being user-friendly and intuitive design, the web extension will provide a diverse and highly accessible solution for users to alleviate the problem of being unable to distinguish between fake and real news articles. This problem is further highlighted by the recent surge in the use of Artificial Intelligence to create fake news articles, further increasing the risk of more users falling prey to malicious parties on the internet.

To summarize, this project aims to highlight the importance of creating a solution to tackle fake news, providing a streamlined solution to users to ensure a safer browsing environment online.

Table of Contents

Chapter 1: Introduction [2 pages]	5
1.1 Project Concept and Motivations	5
1.2 Project Objective	5
1.3 Variables to Determine	6
1.4 Further Research	6
1.5 Fake News Dataset	6
Chapter 2: Literature Review [5 pages]	7
2.1 Introduction to the Domain of Fake News	7
2.1.1 The Necessity of Ensemble Methods in Modern Fake News Detection	7
2.1.2 Limitations to Complex Model Training	8
2.1.3 Identifying the Most Suitable Method for Fake News Detection	9
2.2 Overall Analysis of Literature Review	10
2.3 Research Insights for Project	11
Chapter 3: Project Design [4 pages]	12
3.1 Project Timeline	12
3.2 Project Methodology - Waterfall	13
3.3 Project Design Outline - Solution Flow	13
3.4 Technologies and Libraries	14
3.5 Architecture of Chrome Extension	14
3.6 Project Evaluation	15
Chapter 4: Implementation [3 pages]	16
4.1 Data Loading and Exploration	16
4.2 BERT Model Training	17
4.3 Development and Deployment of Chrome Extension	18
Chapter 5: Evaluation [3 pages]	19
5.1 Evaluation of BERT Model	19
5.2 Evaluation of Fake News Chrome Extension	20
5.3 Further Discussion on the Performance of the Fake News Extension	20
5.3 Project Success, Challenges, and Failures	21
Chapter 6: Conclusion [2 pages]	22
6.1 Project Deliverables	22
6.2 Future Improvements	23
6.3 Project Conclusion	23
References	24

Project Github Link	25
Appendix	26

Chapter 1: Introduction [2 pages]

1.1 Project Concept and Motivations

The Project Template chosen is 'Project Idea Title 1: Fake News Detection' under the CM3060 Natural Language Processing Module.

Fake News consists of news articles or stories that are published online based on falsehood, on the pretense of spreading false information. Sensational headlines, also known as click-bait, generate the most traffic online and further propagate the notion that garnering more clicks is more important than the information that is being presented. This incentivizes a cycle of generating the most exciting stories that generate more user interaction through clicking enticing headlines instead of providing information that accurately informs the user of current affairs. This, however, also introduces significant dangers in the form of scams, false information, inciting public discourse, and even severe crimes such as identity theft or the loss of lives.

The motivations behind starting this project lie begins with the pressing need for an effective and prevalent solution to tackle the issue of fake news. Artificial Intelligence has been on the rise with malicious parties creating clones of famous trusted figures to deceive and spread false information. Today, more and more of these malicious parties 'are tapping sophisticated artificial intelligence (AI) tools to create "deepfake" voice recordings and videos of people, to fool their relatives and friends into transferring money [5]".

1.2 Project Objective

The objective of this project is to

- develop a web extension that enables a vast majority of people online to identify and filter through the news articles that they view.
- Empowers online users with the ability to safeguard themselves from the dangers of fake news, regardless of their online information literacy.
- This web extension analyzes web information in real-time, identifying falsehoods and misleading narratives that the user encounters online.
- The credibility of the information will be clearly presented to the user through clear visual cues in the extension and will allow the user to make better-informed decisions on whether to trust a particular article that has been read online.
- Evaluate the performance of this extension based on real-world news that is up to date.

1.3 Variables to Determine

The first thing to determine would be the most popular browsers that are used online. Figure 1 (In Appendix) indicates that Google Chrome is the most commonly used web browser. This trend is also seen across multiple statistics as seen in figures 2 and 3 (In Appendix). Given that the large majority of users are using Google Chrome, the web extension to be developed as part of the project solution will be based on the Manifest V3 architecture. This means that it will be able to be housed in the Chrome web store, allowing users to easily download and use the extension in a few clicks.

Secondly, the next thing to determine would be to identify the most effective text classification methods and employ them in our solution. Popular text classification methods involve using the following:

1. Naive Bayes: A simple model based on Bayes' theorem and assumes independence across features.
2. Logistic Regression: Utilizes a statistical model that determines the probability of a certain class and their relationships.
3. Decision Trees: Splits data and uses recursion to produce tree-like structures for classification.
4. Random Forest: Uses multiple decision trees to improve accuracy and reduce the chances of overfitting.

This will be further investigated in the Literature review, where the most effective model or combination of models will be highlighted. This will then serve as the backbone for the infrastructure for our solution model when tackling fake news.

1.4 Further Research

A greater depth of research is required to determine the best approach when developing the solution to tackle fake news. I have identified the following factors to consider:

- What are the lapses in existing solutions that should be implemented?
- What are the necessary/recommended hyperparameters to adjust for model tuning?
- What are the evaluation metrics that should be used when involving the domain of fake news?
- What kind of data preprocessing is required?
- What dataset should be used to prepare the model for the solution?

1.5 Fake News Dataset

The dataset to be used should contain text information from articles, each with a separate label of being either 'True' or 'False'. This is the baseline to train accurate models effectively.

Chapter 2: Literature Review [5 pages]

2.1 Introduction to the Domain of Fake News

Fake News is a problem that involves text with false information. This means that modern text classification methods are imperative to distinguish real and fake news, given the number of articles published daily. The following papers discuss text classification models and their relevance to fake news.

2.1.1 The Necessity of Ensemble Methods in Modern Fake News Detection

Paper Title: Fake News Detection Using Machine Learning Ensemble Methods [1]

This paper discusses the performance of several text classification models on fake news. The author evaluated the performance of individual text classification models, alongside the performance of other ensemble models across four different datasets.

From Figure 4 (In Appendix), we can see that the Bagging Classifier, (XGBoost), was the best-performing model across all the different evaluation runs performed on the four different datasets. This model is under the category of ensemble learners, where multiple text classification models are layered and run as opposed to conventionally running a single text model for evaluation. The paper also determined that the accuracy of ensemble learners is 14.65% higher than individual algorithms. This demonstrates the superior capability of ensemble models compared to a single text classification algorithm in terms of accuracy.

However, despite their better accuracy, it is important to note that accuracy alone on a single dataset is not definitive, and making the logical deduction that layered text classification models are always better may not be outright applicable in all cases of detection. This is seen in Figure 4 (In Appendix) when Random Forest scored a better accuracy score than the Bagging Classifier for Data Set 3 but scored an extremely low score for Data Set 2 in comparison. Data Set 2 is notably larger than Data Set 3, containing over 20,000 articles as opposed to slightly over 3,300. This brings up the possibility of certain text classification models performing better or worse depending on the size of the data set used for evaluation.

The average best accuracy was achieved by the Boosting Classifier with XGBoost. While ensemble learners do perform better overall in terms of accuracy, to state that they are outright better than single algorithms, we need to evaluate them on either multiple metrics or several datasets.

To further this point, the author evaluated the models across several other analytical metrics. From Figures 5-8 (In Appendix), these comprehensive comparisons across three other metrics show that it is definitively evident that ensemble learners perform vastly superior to a single algorithm when tackling fake news detection. As the author states, "The ensemble learners

have shown an overall better score on all performance metrics as compared to the individual learners [1]”. Based on these findings, it is clear that running a single algorithm is not sufficient in terms of reliably being able to detect fake news today. This consideration will be taken into developing our solution, where ensemble learners will be employed and tuned to be a part of the main infrastructure that serves as the fake news detection solution.

To conclude his paper, the author noted that machine learning needs to be investigated more thoroughly to gain a deeper understanding of the landscape of fake news detection. Given the recent publishing date, this paper mainly shows how ensemble techniques are necessary for modern fake news detection today.

2.1.2 Limitations to Complex Model Training

Paper Title: A Systematic Literature Review and Meta-Analysis of Studies on Online Fake News Detection [2]

Having established that ensemble models perform better across the board when compared with single algorithms, we also investigate fake news detection approaches more deeply, looking into machine learning usage and its relevance to fake news detection over the years. This paper studies the frequency of the different approaches that can be used for fake news detection over the years.

Based on Figure 9 (In Appendix), the method used for fake news detection is generally split into two main categories, Machine Learning and Deep Learning. This is also supported by the findings in the paper, where the author states “DL and ML emerging as the approaches relied upon most often for fake news detection [2]”. Within each category, Central Neural Networks and Random Forests were the most popular approaches for Deep Learning and Machine Learning respectively. Additionally, the trend of approaches in the previous years in Figure 10 (In Appendix) also shows that Deep Learning and Machine learning are the two most popular approaches in deep learning.

More notably, the author concludes that “the sample size and the accuracy of the fake news detection method are strongly negatively correlated. This underscores how crucial it is to use a large number of samples when testing fake news detection methods [2]”. This is in stark contrast to the results seen in the previous paper, where Random Forest performed worse despite being evaluated on a larger dataset.

In summary, the findings in this paper conclude that despite deep learning being the most common approach, conventional ensemble methods such as Random Forest are just as capable and accurate for the context of fake news detection. Overall, this introduces another important consideration of the project solution, which is the raw computational power and time needed in order to run a Central Neural Network as opposed to running an ensemble model. More testing will be needed to investigate which model is appropriate for the scope of this project.

2.1.3 Identifying the Most Suitable Method for Fake News Detection

Paper Title: A Comparative Study of Machine Learning and Deep Learning Techniques for Fake News Detection [3]

This next paper investigates the performance of Deep Learning against Machine Learning. The importance of this study mainly highlights the performance of several classical text classification models, an ensemble model that combines these models, as well as multiple machine learning models.

In general, it was found that more advanced ensemble methods indeed outperform classical machine learning methods as seen in Figures 11-12 (In Appendix). Despite this overarching conclusion, the base BERT model, which was trained before testing against the other models, turned out to be the best-performing model specifically for a COVID-19 dataset. This performance was in line with the best-performing models despite the lower complexity of the model. This effectively means that ensemble methods do not necessarily require the use of highly advanced Central Neural Networks to be as effective in fake news detection.

Another finding is that while ensemble methods with TF-IDF feature usage do generally outperform the other dataset on the LIAR dataset, the nature of the dataset itself creates an increased difficulty for the models tested to discern real news from fake news as it contains short pieces of text information. The two BERT models performed exceptionally well across the datasets in all tests run, demonstrating the sheer capability of being able to make use of short textual information to the full extent.

The author then concludes that “the experimental results indicate that no single technique can deliver the best performance scores across all used datasets [3]”. As such, we can conclude that Deep Learning which uses ensemble learning is more than sufficient when it comes to fake news detection alone. To conclude, we will choose BERT models as our primary approach.

2.1.4 A Further Study into BERT-based Models

Paper Title: Predictive Intelligence in Harmful News Identification by BERT-based Ensemble Learning Model with Text Sentiment Analysis [4]

This paper studies the BERT model at sophisticated levels, focusing on its performance alone in both harmful news and fake news detection. The study achieved a near-perfect result for fake news detection with the use of a BERT model. These findings also share the similar sentiments found in Paper 3 where different advanced BERT models were found to have an outstanding capability to make use of limited text information given to detect fake news. However, this necessitates very high computational requirements with additional time for training which will impede the development of a widespread accessible solution.

Paper Title: Fake News Detection Using Enhanced BERT [5]

This paper determined that “pre-trained algorithms are more effective at identifying fake news because it takes less time to train them and yields better results [5]”. BERT models that have been given further training and tuning will perform better than pre-trained models. However, the process of training is usually time-consuming and improves an already effective model. This can also be seen in paper [4] where the proposed method improved upon the BERT model where “the F1-score in the test set is improved by 5.7%. [4]”.

Paper Title: Fake News Detection Using BERT Model with Joint Learning [6]

The previous finding that using pre-trained models is more effective at fake news detection is also supported in this paper which determined that a pre-trained BERT model was competitive at fake news classification when assessed against BERT models that use a joint approach based on relational feature classification and named entity recognition. A slight advantage was seen in the default FakeBERT model in Figure 13 (In Appendix) in its predictive performance. Generally, all models performed exceptionally well when detecting fake news with the model tuned in the paper performing expectedly better when tuned for a dataset with a smaller number of news subjects.

As such, BERT models, highly advanced or pre-trained, perform exceptionally well for fake news detection when lined up against older conventional methods of fake news detection.

2.2 Overall Analysis of Literature Review

To summarize my findings,

- Ensemble methods, which comprise multiple text classification algorithms, outperform conventional text classification methods that only involve a basic algorithm. We will utilize BERT models as part of our Google Chrome extension
- The process of improving a BERT model that caters to specific datasets will yield better predictive results after training and tuning of hyperparameters. However, this requires a lot of time and computational hardware.
- Greater raw computational power does not equate to better results for fake news detection. Baseline BERT models have found success despite having a lower technical complexity compared to more advanced state-of-the-art models (CNN models) seen today. These are often ready to deploy and implement for use in fake news detection solutions.

2.3 Research Insights for Project

Linking these findings back to the research questions set for our project:

- What are the lapses in existing solutions that should be implemented?

The solution must include the ensemble method for our solution model. Conventional text classification models are no longer sufficient in filtering out fake news, especially with AI and newer methods being used in research testing. The BERT model and Random Forest will be explored as part of early testing, with the BERT model serving as the main architecture for our fake news solution.

The BERT model performs exceptionally well across multiple datasets of different news categories even without further training or tuning. This is a viable approach that can be used for the detection of fake news in the Google Chrome Extension and will not necessarily require training to be effective.

- What are the necessary/recommended hyperparameters to adjust for model tuning?
- What dataset should be used to prepare the model for the solution?
- What kind of data preprocessing is required?

For the BERT model, the dataset will be split into a training set and a validation set to further improve the model. The hyperparameters tuned will include the batch size of text input and the epoch for the different training cycles. A fake news dataset with labeled text to indicate if it is fake or real will be used. The dataset used must be of a sufficient size and prior training can be utilized to increase the performance of certain text classification models. The dataset should be cleaned appropriately and even multiple or split datasets can be used to evaluate the performance of our models.

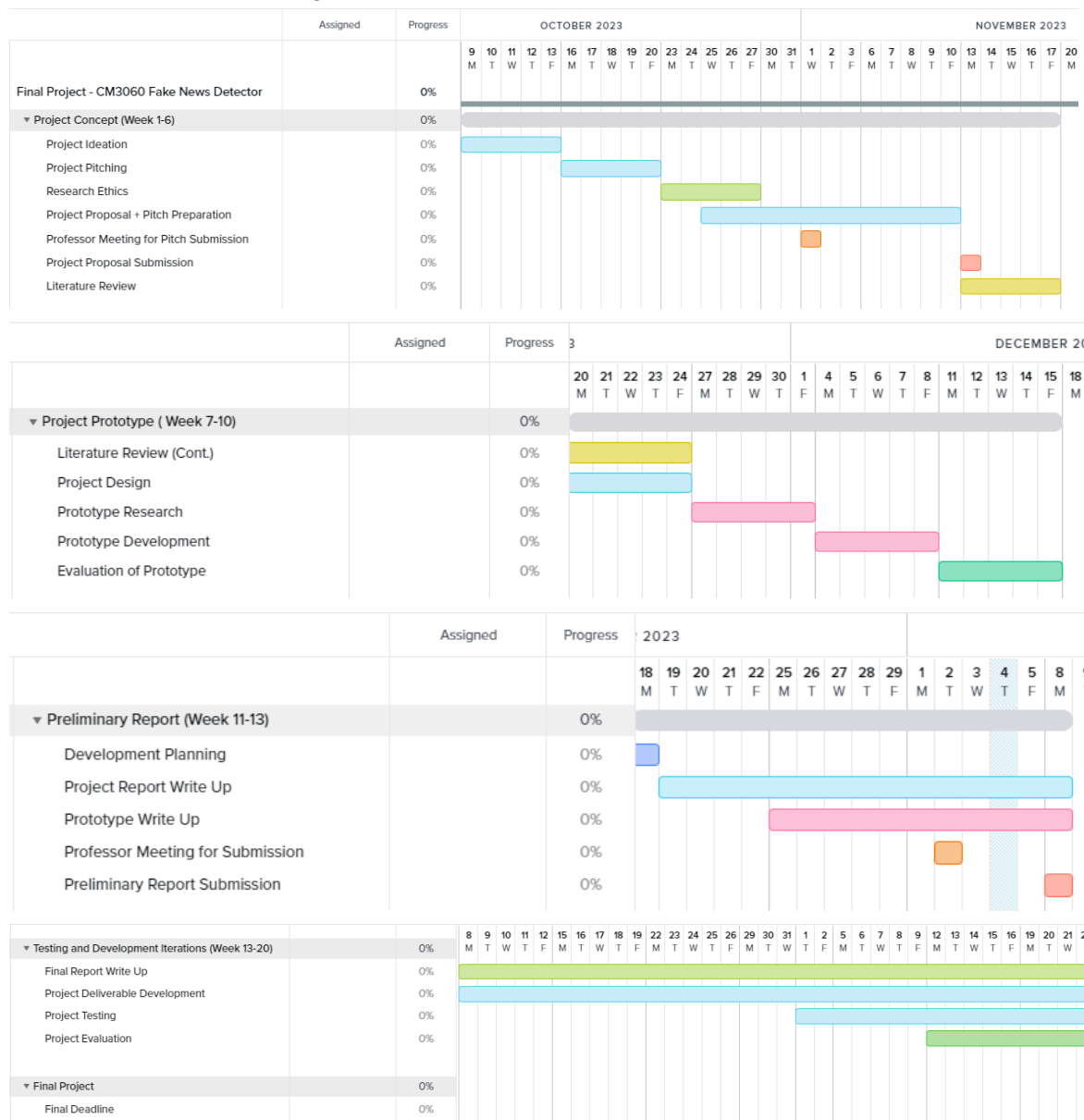
- What are the evaluation metrics that should be used when involving the domain of fake news?

When looking into including the use of a CNN in the text classification model, considerations such as the computational strain must be taken into consideration as ensemble ML methods perform just as well without the increased computational power required. This is important as the aim of the project is to ensure the vast majority of users can use the web extension easily. We do not want to introduce the problem of having high computational demands for the user depending on the implementation of the web extension. As such, we will stick to the BERT model for the solution and use this text model for the project solution. We will determine its performance using metrics such as accuracy, precision, and f1-score as well as determine the training loss compared to the validation loss.

Chapter 3: Project Design [4 pages]

3.1 Project Timeline

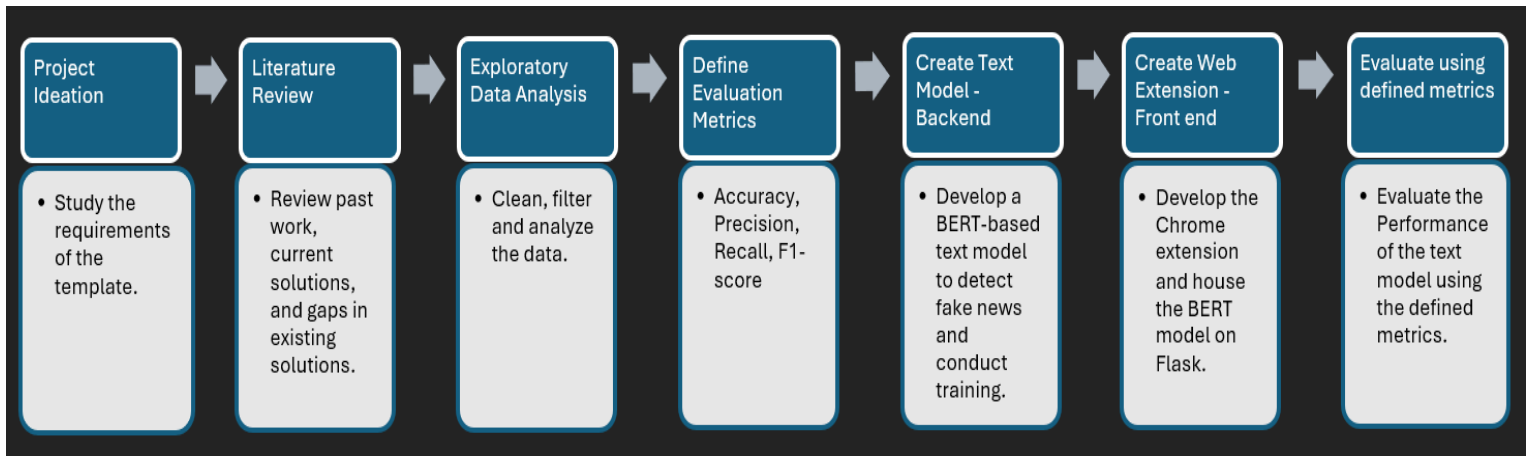
This project consists of first creating a text-classification model that is appropriate for use against the problem of fake news. This model is then deployed into a backend with a front-end extension that allows the user to use the model on their browser. The general project plan will be outlined in the following Gantt Chart:



As seen above, the work schedule planned for the following weeks generally follows the pace outlined as per the weeks in Coursera. Additionally, regular meetings with the SIM professor will be scheduled to ensure that the project remains in focus in line with the requirements.

3.2 Project Methodology - Waterfall

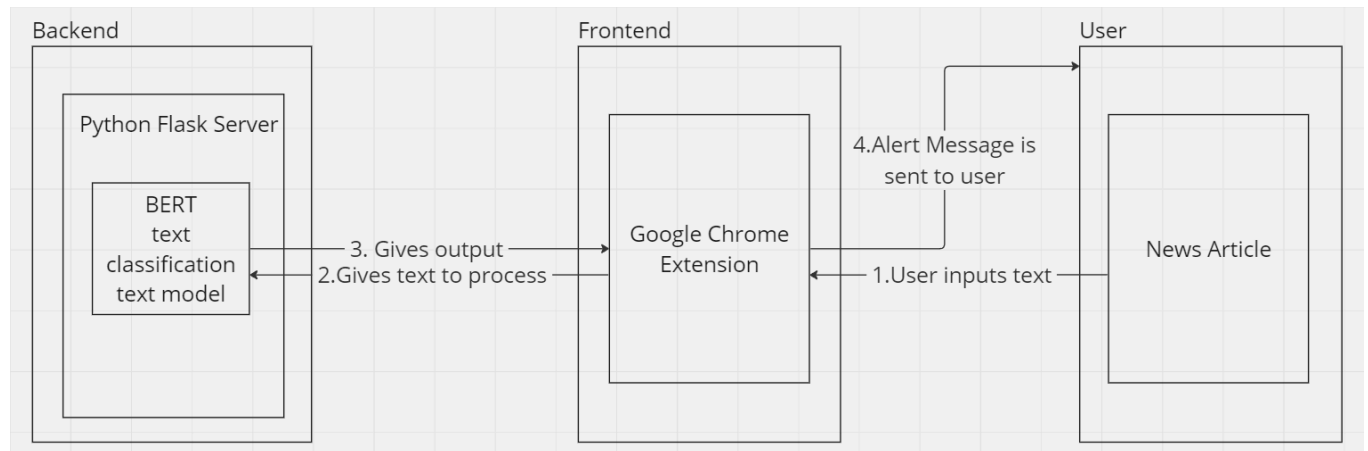
The diagram above will outline the accent process of the project, with the architecture of the solution seen below:



Testing and evaluation will be conducted after the development of the BERT model for the Chrome extension. From the literature review findings, based on the performance of the BERT model developed with present hardware constraints, a pre-trained model will be used for the Chrome extension if the results are not satisfactory.

3.3 Project Design Outline - Solution Flow

The architecture of the solution in the form of the Google Chrome Extension is seen below:

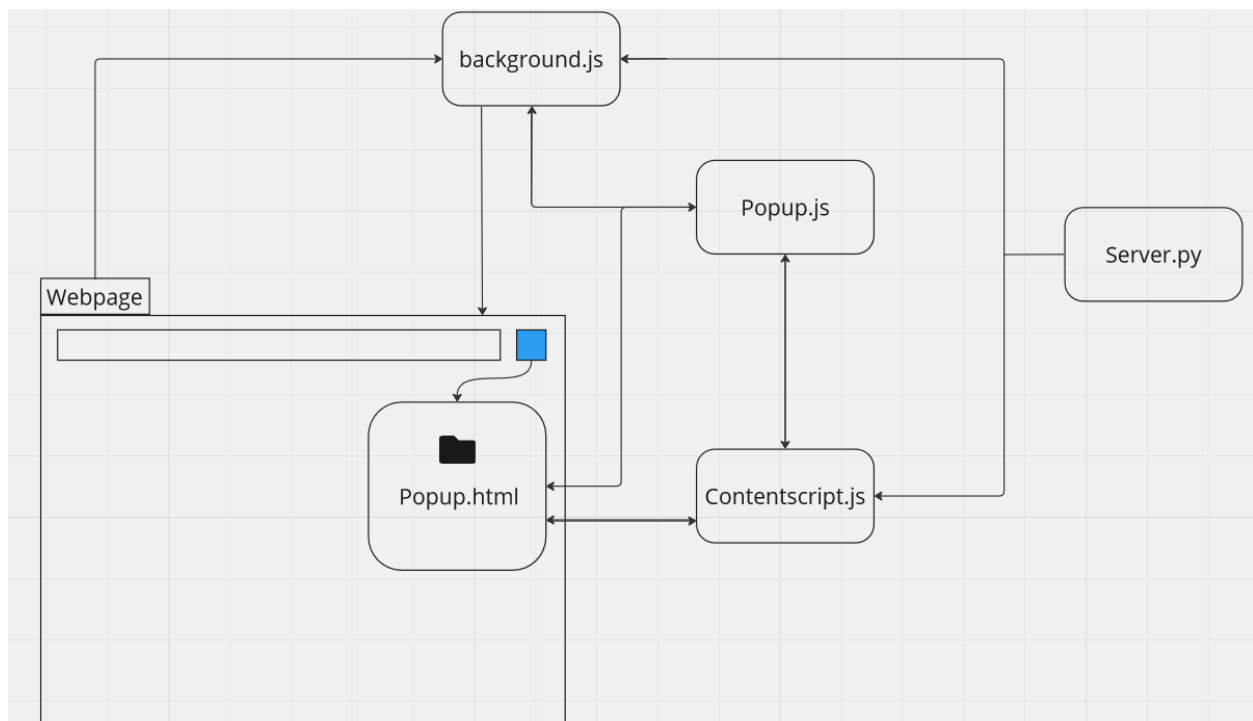


3.4 Technologies and Libraries

The first prototype was developed on Jupyter Notebook, using imported libraries to run different text classification models on a CSV dataset containing fake news.

The backend developed further down the line will be running on Python Flask that communicates with a Chrome extension powered on the Manifest V3 architecture. The architecture is outlined in a basic diagram in the next section.

3.5 Architecture of Chrome Extension



In summary, the background and content script await the user to input text via the popup.js to send to the BERT model housed on the flask server, server.py. The BERT model then evaluates the text and returns an output of either safe or not safe. This is then reflected to the user by a message that appears on the popup.html. Based on the findings from the BERT model, an alert message will inform the user whether the news is predicted to be fake or safe for consumption.

3.6 Project Evaluation

As seen in the studies conducted for most text classification studies, the metrics used will include:

1. Accuracy
 - Measures the general correctness of the text classification model.
2. Precision
 - It is a measure of the accuracy of positive predictions.
3. Recall
 - Measures the ability to identify all positive instances in the dataset.
4. F1-Score
 - Harmonic mean of precision and recall.

The use of a confusion matrix will also provide a better visualization of the performance. Additionally, the training loss and validation loss will be plotted to determine if training and improving the model for deployment in our Chrome extension is feasible given the time and technological constraints.

A survey conducted using Google Survey will be used to qualitatively evaluate the ease of use of the fake news extension.

4.2 BERT Model Training

```
# Shuffle the combined dataset
combined_data = combined_data.sample(frac=1).reset_index(drop=True)

# Split the dataset into training and validation sets
train_size = int(0.8 * len(combined_data))
val_size = len(combined_data) - train_size

train_data = combined_data[:train_size]
val_data = combined_data[train_size:]
```

The next step involves training the BERT model by splitting the dataset into a validation and training set.

```
for step, batch in enumerate(train_dataloader):

    if step % 40 == 0 and not step == 0:
        elapsed = format_time(time.time() - t0)
        print(' Batch {:>5,} of {:>5,}. Elapsed: {:.format(step, len(train_dataloader), elapsed))

    b_input_ids = batch[0].to(device)
    b_input_mask = batch[1].to(device)
    b_labels = batch[2].to(device)

    model.zero_grad()

    outputs = model(b_input_ids,
                    attention_mask=b_input_mask,
                    labels=b_labels)

    loss = outputs.loss

    total_train_loss += loss.item()

    loss.backward()

    torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)

    optimizer.step()

    scheduler.step()

avg_train_loss = total_train_loss / len(train_dataloader)
train_loss_values.append(avg_train_loss)
```

Based on the findings from the literature review, the BERT model was tuned such that training, weights such as the epoch values, batch size, and total number of data entries were feasible with personal hardware limitations. After each iteration of training, the average loss of training and validation were recorded and saved for evaluation further down the project timeline(Figure 16).

```
Accuracy: 0.5500
Precision: 1.0000
Recall: 0.4000
F1-score: 0.5714
```

After tuning, multiple runs of training and adjustments were conducted to obtain the best possible training results. However, the results of the training were barely satisfactory as the BERT model requires a large number of data entries and time to be able to be trained to the point where the performance is reliable for fake news identification.

```

import torch
from transformers import BertForSequenceClassification, BertTokenizer
import pickle
# Load BERT model and tokenizer
model = BertForSequenceClassification.from_pretrained('bert-base-uncased')
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Save the model and tokenizer
model_path = './model/bert_model.pkl'
tokenizer_path = './model/tokenizer.pkl'

# Save the model using torch.save()
torch.save(model.state_dict(), model_path)

# Save the tokenizer using pickle
with open(tokenizer_path, 'wb') as f:
    pickle.dump(tokenizer, f)

```

Afterward, the decision was made to save the current model and its corresponding weights and move on to developing the Chrome Extension before coming back to improve the model. A pre-trained BERT model will first serve as the main fake news identification service in our Chrome extension, which has been proven to be just as robust and competitive as trained BERT models.

4.3 Development and Deployment of Chrome Extension

The fake news Chrome extension operates on the latest manifest V3 architecture from Google. This extension works together with a Python Flask server(Figure 18) that initiates and houses the BERT model, which evaluates text sent from the user through an input box(Figure 19).

Breaking it down, the Chrome extension consists of a popup.html which comes in the form of a popup window for the user with a 'Check for fake news' button. When the user inputs text and clicks the button, the event listener in popup.js is triggered. This sends a message to the content_script.js to retrieve the text and sends a request to the Flask /predict endpoint(Figure 20).

The selected text is then processed, printing an output depending on whether it is predicted to be fake or real news(Figure 21). This result is returned as a JSON response. Lastly, based on the response output from the BERT model, an alert message is presented to the user in the browser(Figure 22).

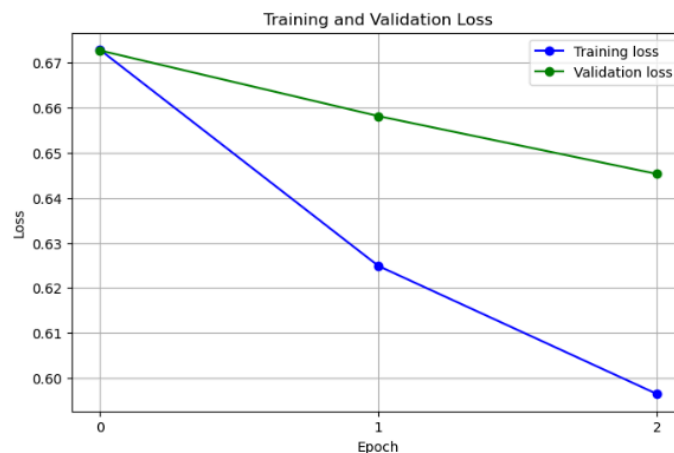
Chapter 5: Evaluation [3 pages]

5.1 Evaluation of BERT Model

The BERT model was trained using different batch sizes and a standard value of epoch = 3. The training loss and validation loss were plotted for each of the test cycles conducted.

Batch Size	Epoch	Evaluation Metric Results
20	3	Accuracy: 0.5500 Precision: 1.0000 Recall: 0.4000 F1-score: 0.5714
50	3	Accuracy: 0.8500 Precision: 0.9091 Recall: 0.8333 F1-score: 0.8696

From the evaluation scores for the BERT model above, we see a trend of the model improving across all metrics after the BERT model has been given a greater number of data entries to evaluate and train on. This is consistent with the literature review findings where the performance of the model is linear to the size of the training set of data.



Lastly, plotting the training loss and validation loss shows a trend in which both losses decrease over the iterations of the epoch. The decrease in training indicates that the model is learning and improving the ability to classify fake news. Similarly, the decrease in validation loss is indicative of the model improving on the unseen validation data as well. Despite the hardware limitations and long training time of three hours per cycle, additional training for text models continues to improve their performance based on the dataset used.

5.2 Evaluation of Fake News Chrome Extension

A Google form(Figures 23-25) was used to collect feedback related to the ease of deployment and use of the Google Chrome fake news extension. In general, the extension built was generally found to be easy to add to the browsers of users with most users being able to use the extension once the Python server was initiated.

Certain users who did not have Python previously installed faced difficulty adding the extension to their Chrome browser. This also links to setting up the Python Flask server which some users were not able to do by themselves. Among this group of users, the most common feedback was to integrate the entirety of the solution onto the Google Chrome extension file loaded into the extension page.

Users also submitted areas of improvement in an open-ended question for further developments of the Chrome Extension (Figure 26).

The reliability of the Chrome extension itself was not developed sufficiently to the point where the extension was able to reliably detect fake news. This one aspect of the project suffered given the time constraints to develop and train a BERT model, followed by developing a Chrome Extension to work with the Python Flask server from scratch.

5.3 Further Discussion on the Performance of the Fake News Extension

The use of a default pre-trained model led to varying results based on the type of text input given to the model. This is due to the limited exposure that the default BERT model has and is not able to accurately determine the validity of the news text input at a proficient level. The following table provides a general overview of its performance based on testing with several users who responded in the Google Form.

Text inputs from news	Genre	Pass	Fail	Reliability
34	Political News	14	20	41%
34	Global Headlines	15	19	44%
34	Business	9	25	26%

As seen from the results, the Chrome extension using the default BERT model is not presently able to discern fake news accurately. However, it does provide a baseline level of protection against the possible consumption of fake news. Given more time and computational power, the BERT model would be trained using larger batch sizes over greater epochs and several genres of news datasets. This would allow the deployment of a diverse BERT model and consequently Chrome extension that would be able to tackle fake news better.

5.3 Project Success, Challenges, and Failures

This project succeeded in several aspects outlined in the project objectives in the beginning. A BERT text classification model was successfully built and trained using Jupyter Notebook. The architecture of the solution presented before the development of the Chrome Extension was also successfully built, working together with a Python Flask server.

By the end of this project, the final developed Google Chrome extension provided users with a widespread and accessible solution to safeguard against fake news. This solution is vast in terms of its applicability to users and analyzes news text in real-time using a BERT model housed on a Flask server running in the background. The credibility of the news is presented clearly in the form of a warning message to the user, informing the user if the current news article being viewed should be trusted. Given more time, the functionality to add relevant timely updates to the BERT model would be explored. This function would allow the processed news to be fact-checked based on the latest updates to the BERT model.

The first challenge encountered during this project would be the hardware limitations when training and building the BERT model. During debugging, a minimum hour is required to test if the BERT model will be able to run after changes to the training settings. This issue is further compounded when needing to run multiple iterations of training, testing, and optimization. As such, the final BERT model is not at its optimal state and was not trained to my satisfactory level. In the end, a pre-trained BERT model was used during the building of the Google Chrome extension and integrated with the Flask server. This compromise was needed in order to ensure that the solution as a whole would be complete within the given time constraints.

Due to the limited time constraints, I was not able to incorporate the saved weights of my BERT model after running multiple rounds of training into the Chrome Extension. This problem further extends to the performance of the Google Chrome extension itself, where the performance of evaluating fake news did not achieve a reliable level. I did not manage to determine the cause of why the BERT model would sometimes return a warning message against reliable news sources and vice versa. The absence of errors in the terminal or Chrome extension debugging window worsened this problem, making it extremely difficult to pinpoint the root of the performance issue. The process of adding the Chrome extension itself also required users to already have Python installed. This issue was overlooked and only discovered when users without Python installed were asked to participate in the Google Form testing.

In hindsight, I should have developed a smaller version of the extension that evaluates news text using a trained BERT model and is housed within the Google Chrome extension itself without the need to initiate a separate Python server. This would solve the issue of some users needing to install Python to be able to use the Chrome extension. A lot of time was spent learning the newest Manifest V3 architecture from Google to develop a working extension for future compatibility. This lengthy process did shorten the time for development in other areas of the project but served as a valuable learning experience in modern extension development.

Chapter 6: Conclusion [2 pages]

6.1 Project Deliverables

Linking back to the project objectives,

- Objective 1: Develop a web extension that enables a vast majority of people online to identify and filter through the news articles that they view.
- Project Outcome: A web extension has been built on Google Chrome that allows online users to add a fake news detector into their browser easily.
- Objective 2: Empower online users with the ability to safeguard themselves from the dangers of fake news, regardless of their online information literacy.
- Project Outcome: A text input function has been added that allows text obtained from news to be easily verified with a few button presses using an advanced BERT model.
- Objective 3: This web extension analyzes web information in real time, identifying falsehoods and misleading narratives that the user encounters online.
- Project Outcome: The extension developed actively communicates with the text classification model and gives warnings that correspond to live predictions. These warnings allow news on the internet to undergo another level of validation for checking.
- Objective 4: The credibility of the information will be clearly presented to the user through clear visual cues in the extension and will allow the user to make better-informed decisions on whether to trust a particular article that has been read online.
- Project Outcome: A popup message is presented to the user after the model returns an outcome based on its prediction. This message warns the user in the event of a possible presence of fake news.
- Objective 5: Evaluate the performance of this extension based on real-world news that is up to date.
- Project Outcome: The BERT model only evaluates news text based on prior training.

In general, the large majority of the project objectives have been met with the main goal of empowering Google Chrome users with a fake news detector being completed.

6.2 Future Improvements

Given the opportunity, here is a list of tasks I would continue to work on:

1. Incorporate the saved weights of the train BERT model into the Chrome Extension
2. Look into the possibility of adding the BERT model into the Chrome Extension without the need for a Python Server
3. Improve the performance of the BERT model, and subsequently the Chrome Extension.
4. Add the function to automatically evaluate the <h1> content of the current webpage and display its reliability to the user once the extension button is pressed.
5. Optimize the training process of the BERT model such that it takes less time and increase the number of epoch and batch size for better results.
6. Explore the function to add real-time learning for the model, training the BERT model based on the news websites the user visits.

6.3 Project Conclusion

As this project comes to an end, I strongly believe that this project has successfully achieved developing a strong foundation for a robust fake news detection system. It is a widespread and accessible solution that users can use to protect themselves against fake news.

Despite the challenges faced during the progression of the project timeline, I was able to work navigate through the challenges of debugging, picking up new documentation from scratch and deploying it shortly, as well as learning how to develop a fully functioning fake news detections solution that integrates a front and back end. It has been an incredibly valuable experience in picking up new languages and programming techniques through this journey.

To conclude, this Chrome extension has seen generally positive feedback from users and I am sure that a robust solution will be developed given enough time and computational power. This would ensure that users will be protected from fake news all around the world, across different locations and browsers.

References

- [1] Iftikhar Ahmad, Muhammad Yousaf, Suhail Yousaf, and Muhammad Ovais Ahmad. 2020. Fake News Detection Using Machine Learning Ensemble Methods. *Complexity* 2020, (October 2020), 1–11. DOI:<https://doi.org/10.1155/2020/8885861>
- [2] Robyn C Thompson, Seena Joseph, and Timothy T Adeliyi. 2022. A Systematic Literature Review and Meta-Analysis of Studies on Online Fake News Detection. *Information* 13, 11 (November 2022), 527–527. DOI:<https://doi.org/10.3390/info13110527>
- [3] Jawaher Alghamdi, Yuqing Lin, and Suhuai Luo. 2022. A Comparative Study of Machine Learning and Deep Learning Techniques for Fake News Detection. *Information* 13, 12 (December 2022), 576–576. DOI:<https://doi.org/10.3390/info13120576>
- [4] Szu-Yin Lin, Yun-Ching Kung, and Fang-Yie Leu. 2022. Predictive intelligence in harmful news identification by BERT-based ensemble learning model with text sentiment analysis. *Information Processing and Management* 59, 2 (March 2022), 102872–102872. DOI:<https://doi.org/10.1016/j.ipm.2022.102872>
- [5] Shadi A Aljawarneh and Safa Ahmad Swedat. 2022. Fake News Detection Using Enhanced BERT. *IEEE Transactions on Computational Social Systems* (January 2022), 1–8. DOI:<https://doi.org/10.1109/tcss.2022.3223786>
- [6] Wesam Shishah. 2021. Fake News Detection Using BERT Model with Joint Learning.: EBSCOhost. *Ebscohost.com*. Retrieved March 1, 2024 from <https://web.p.ebscohost.com/ehost/pdfviewer/pdfviewer?vid=0&sid=48b70765-c331-4578-9f69-5bf470e90b41%40redis>
- [7] Nadine Chua. 2023. Scammers use deepfakes to create voice recordings and videos to trick victims' family, friends. *The Straits Times*. Retrieved March 1, 2024 from <https://www.straitstimes.com/singapore/scammers-use-deepfakes-to-create-voice-recordings-and-videos-of-victims-family-friends-to-trick-them>
- [8] Oberlo. 2023. Most Popular Web Browsers in 2023 [Dec '23 Update] | Oberlo. *Oberlo.com*. Retrieved March 1, 2024 from <https://www.oberlo.com/statistics/browser-market-share#:~:text=As%20of%20November%202023%2C%20Google%27s,Chrome%20to%20browse%20the%20internet>
- [9] Similarweb. 2024. Similarweb. *Similarweb*. Retrieved March 1, 2024 from <https://www.similarweb.com/browsers/>
- [10] Vennage. 2023. Most Popular Web Browsers In 2023 [Infographic] - Venngage. *Venngage*. Retrieved March 1, 2024 from <https://venngage.com/blog/most-popular-browsers/>

Project Github Link

All relevant work has been uploaded onto a public repository, available here:

<https://github.com/LJCAIston/FYPAIston>

Appendix

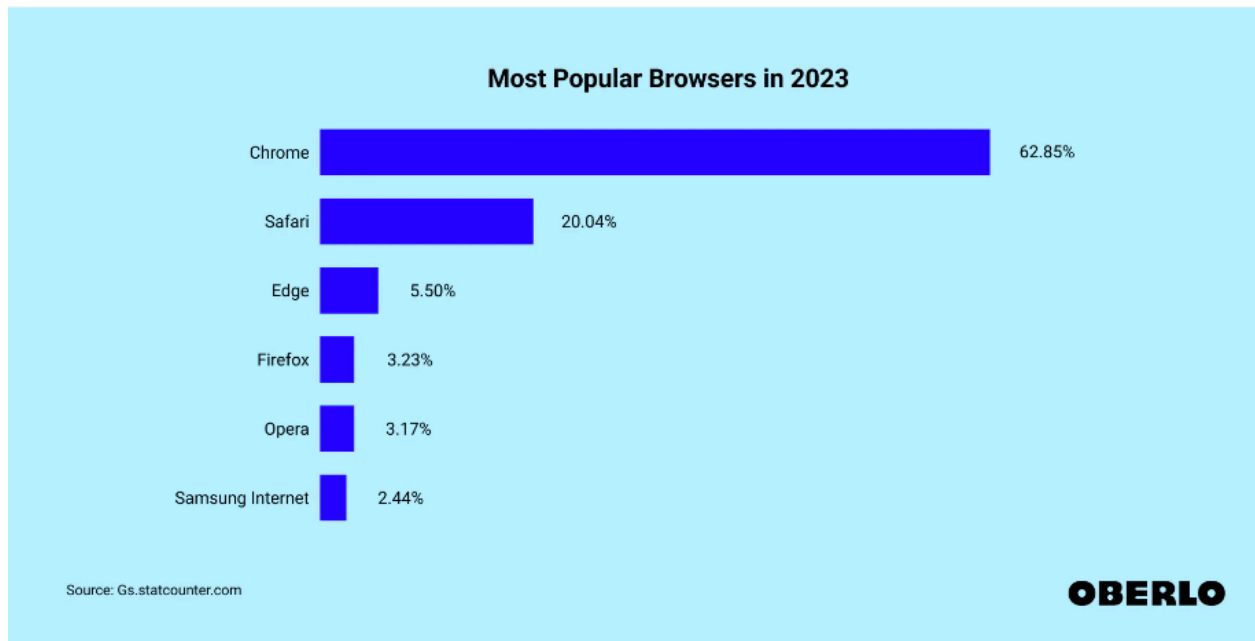


Figure 1. Most Popular Browsers in 2023 [8]

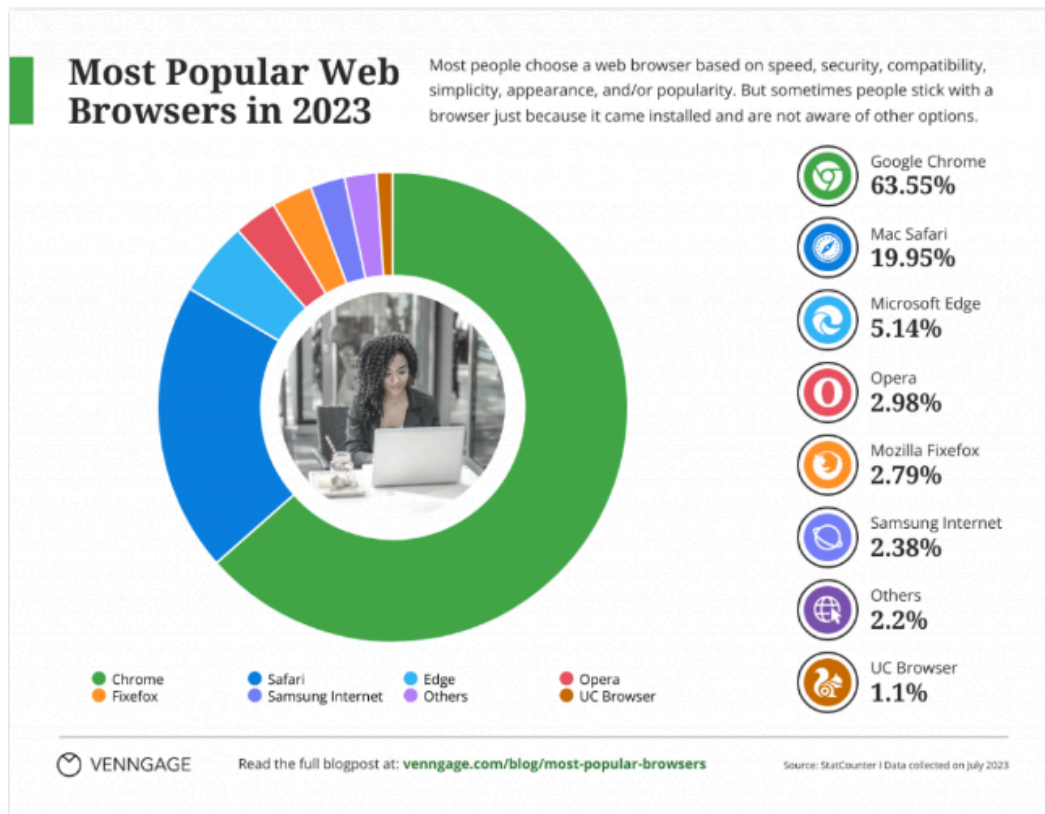


Figure 2. Most Popular Web Browsers in 2023 [10]

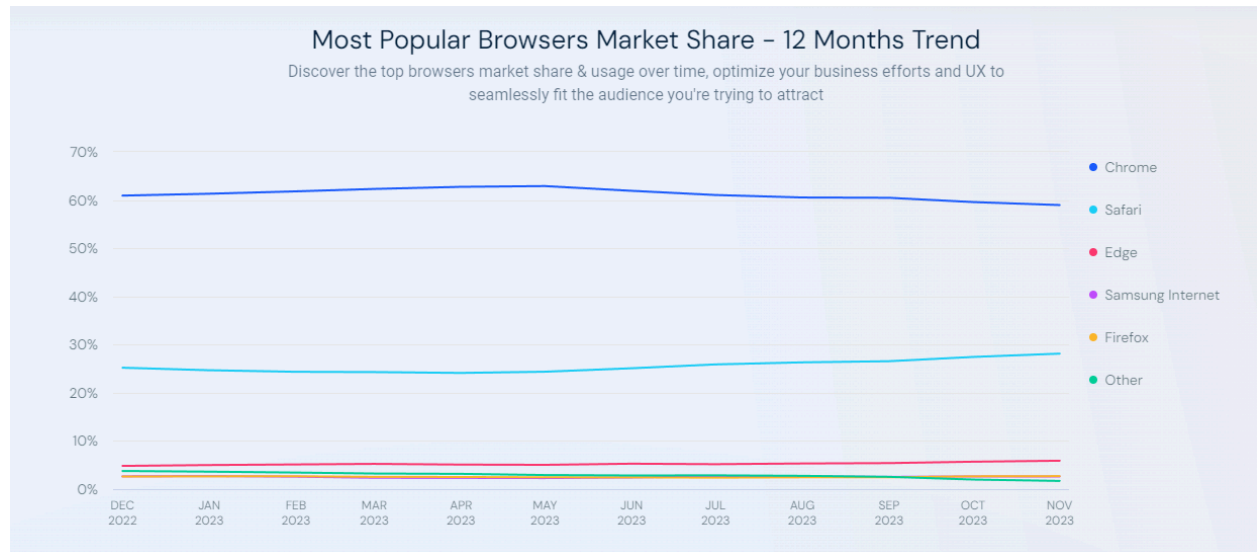


Figure 3. Most Popular Browsers Market Share -12 Months Trend [9]

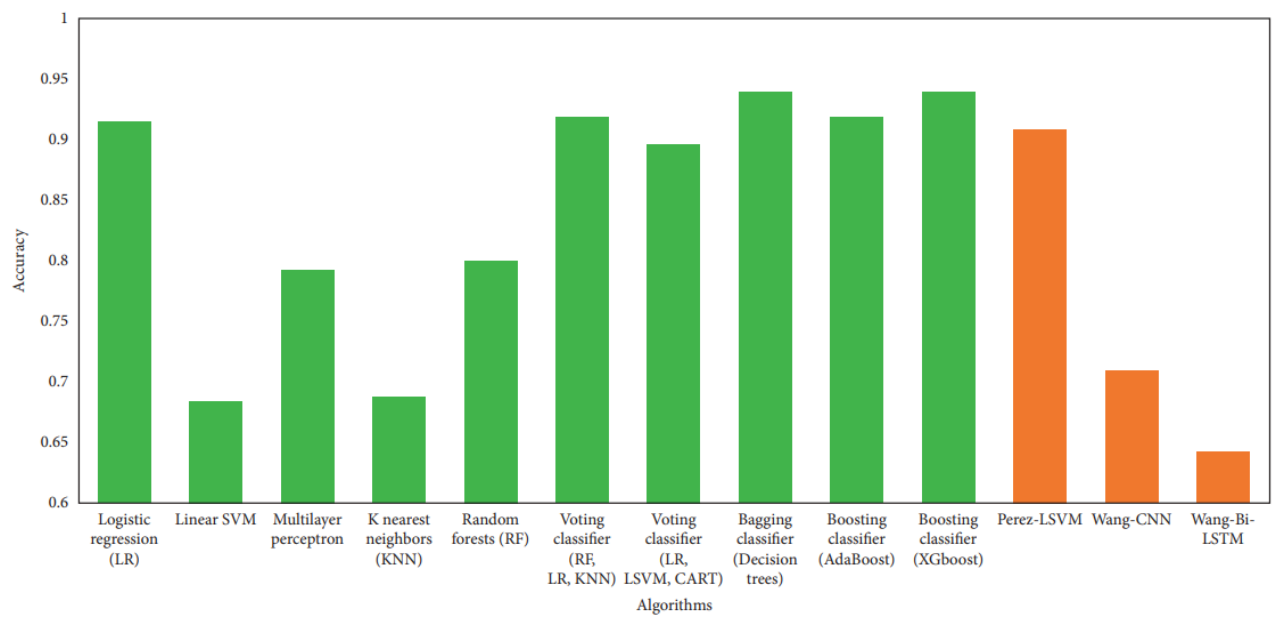


Figure 4. Average accuracy over all datasets [1]

TABLE 3: Precision on the 4 datasets.

	DS1	DS2	DS3	DS4
Logistic regression (LR)	0.98	0.92	0.93	0.88
Linear SVM (LSVM)	0.98	0.31	0.54	0.88
Multilayer perceptron	0.97	0.32	0.93	0.92
K-nearest neighbors (KNN)	0.91	0.22	0.85	0.8
<i>Ensemble learners</i>				
Random forest (RF)	0.99	0.3	0.98	0.92
Voting classifier (RF, LR, KNN)	0.96	0.88	0.92	0.86
Voting classifier (LR, LSVM, CART)	0.94	0.86	0.88	0.83
Bagging classifier (decision trees)	0.98	0.94	0.93	0.9
Boosting classifier (AdaBoost)	0.98	0.92	0.92	0.86
Boosting classifier (XGBoost)	0.99	0.94	0.96	0.92
<i>Benchmark algorithms</i>				
Perez-LSVM	0.99	0.79	0.96	0.9
Wang-CNN	0.84	0.65	0.48	0.72
Wang-Bi-LSTM	0.92	0.43	0.5	0.65

Figure 5. Precision on the 4 datasets. [1]

TABLE 4: Recall on the 4 datasets.

	DS1	DS2	DS3	DS4
Logistic regression (LR)	0.98	0.9	0.92	0.86
Linear SVM (LSVM)	0.98	0.32	1	0.86
Multilayer perceptron	1	0.36	0.96	0.88
K-nearest neighbors (KNN)	0.87	0.24	0.81	0.74
<i>Ensemble learners</i>				
Random forest (RF)	1	0.34	0.93	0.91
Voting classifier (RF, LR, KNN)	0.97	0.89	0.96	0.9
Voting classifier (LR, LSVM, CART)	0.97	0.87	0.96	0.89
Bagging classifier (decision trees)	0.97	0.95	0.94	0.91
Boosting classifier (AdaBoost)	0.98	0.93	0.92	0.86
Boosting classifier (XGBoost)	0.99	0.94	0.94	0.89
<i>Benchmark algorithms</i>				
Perez-LSVM	0.99	0.81	0.97	0.91
Wang-CNN	0.9	0.71	0.29	0.75
Wang-Bi-LSTM	0.78	0.59	0.35	0.61

Figure 6. Recall on the 4 datasets. [1]

TABLE 5: F1-score on the 4 datasets.

	DS1	DS2	DS3	DS4
Logistic regression (LR)	0.98	0.91	0.92	0.87
Linear SVM (LSVM)	0.98	0.32	0.7	0.87
Multilayer perceptron	0.98	0.34	0.95	0.9
K-nearest neighbors (KNN)	0.89	0.23	0.83	0.77
<i>Ensemble learners</i>				
Random forest (RF)	0.99	0.32	0.95	0.91
Voting classifier (RF, LR, KNN)	0.97	0.88	0.94	0.88
Voting classifier (LR, LSVM, CART)	0.96	0.86	0.92	0.86
Bagging classifier (decision trees)	0.98	0.94	0.94	0.9
Boosting classifier (AdaBoost)	0.98	0.92	0.92	0.86
Boosting classifier (XGBoost)	0.99	0.94	0.95	0.9
<i>Benchmark algorithms</i>				
Perez-LSVM	0.99	0.8	0.96	0.9
Wang-CNN	0.87	0.67	0.31	0.73
Wang-Bi-LSTM	0.84	0.44	0.35	0.57

Figure 7. F1-score on the 4 datasets. [1]

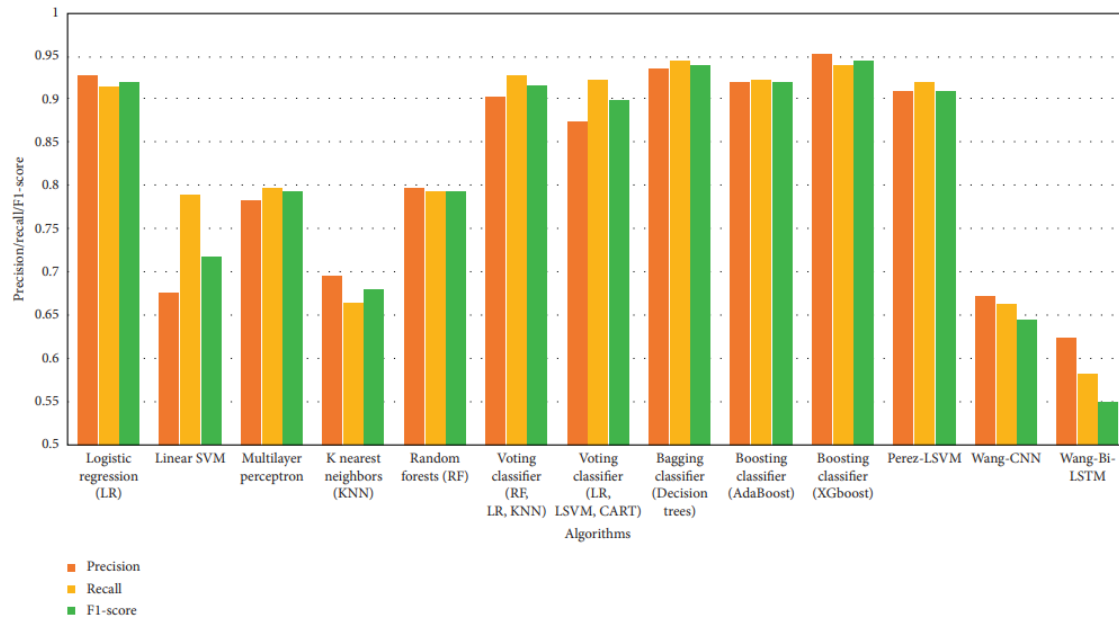


FIGURE 3: Precision, recall, and F1-score over all datasets.

Figure 8. Precision, recall, and F1-score over all datasets. [1]

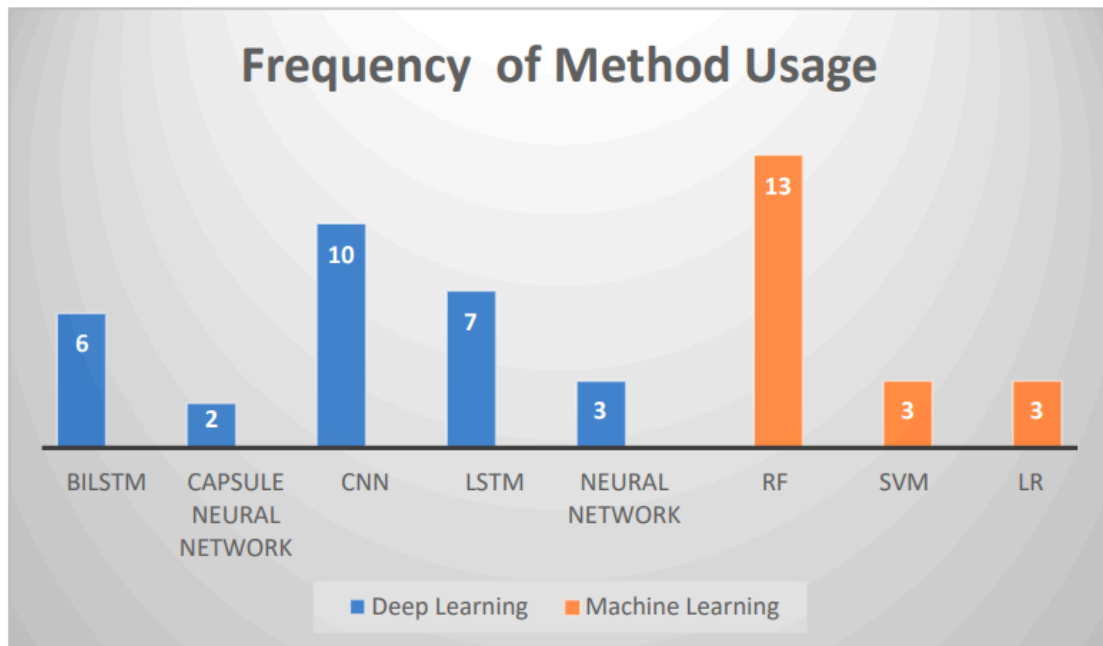


Figure 9. Frequency of method usage.

Figure 9: Frequency of method usage. [2]

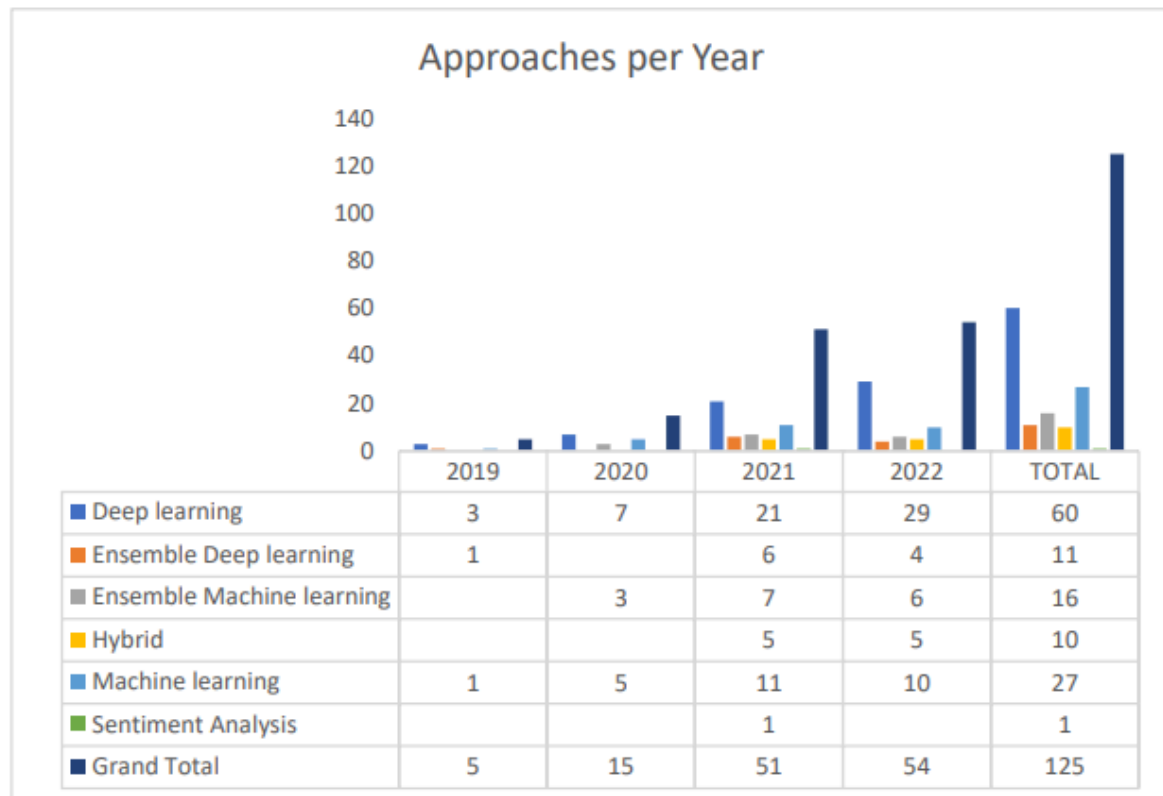


Figure 7. Publications by year.

Figure 10: Publications by year. [2]

Table 13. Fake news detection using classical ML classification models on COVID-19 dataset.

		Dataset			
		COVID-19			
Model	Feature	A (%)	P (%)	R (%)	F1 (%)
LR	CV	0.9313	0.9363	0.9321	0.9342
SVM	CV	0.9318	0.9333	0.9366	0.9349
MNB	CV	0.9051	0.9127	0.9054	0.9090
DT	CV	0.8874	0.8921	0.8929	0.8925
RF	CV	0.9271	0.9221	0.9402	0.9310
XGB	CV	0.8944	0.8942	0.9054	0.8997
Ensemble	CV	0.9327	0.9444	0.9259	0.9351
LR	TFIDF	0.9294	0.9209	0.9464	0.9335
SVM	TFIDF	0.9430	0.9354	0.9571	0.9462
MNB	TFIDF	0.9187	0.9085	0.9393	0.9236
DT	TFIDF	0.8696	0.8778	0.8723	0.8751
RF	TFIDF	0.8654	0.8467	0.9071	0.8759
XGB	TFIDF	0.8883	0.8930	0.8938	0.8934
Ensemble	TFIDF	0.9360	0.9408	0.9366	0.9387
LR	Word2Vec	0.9028	0.9050	0.9098	0.9074
SVM	Word2Vec	0.9051	0.9018	0.9187	0.9102
MNB	Word2Vec	0.8388	0.8083	0.9071	0.8549
DT	Word2Vec	0.7916	0.7993	0.8036	0.8014
RF	Word2Vec	0.8734	0.8794	0.8786	0.8790
XGB	Word2Vec	0.8916	0.8943	0.8991	0.8967
Ensemble	Word2Vec	0.9023	0.9175	0.8938	0.9055
LR	GloVe	0.8182	0.8170	0.8411	0.8289
SVM	GloVe	0.8173	0.8145	0.8429	0.8284
MN	GloVe	0.6486	0.6024	0.6961	0.7421
DT	GloVe	0.7379	0.7450	0.7589	0.7519
RF	GloVe	0.8280	0.8144	0.8696	0.8411
XGB	GloVe	0.8416	0.8381	0.8643	0.8510
Ensemble	GloVe	0.8449	0.8493	0.8554	0.8523

Figure 11: Fake news detection using classical ML classification models on COVID-19 dataset.
[3]

Table 14. Fake news detection using advanced ML and DL classification models on COVID-19 dataset.

Model	Feature	Dataset			
		COVID-19			
		A (%)	P (%)	R (%)	F1 (%)
CNN	Word2Vec	0.9187	0.9022	0.9473	0.9242
BiLSTM	Word2Vec	0.9257	0.9309	0.9268	0.9289
BiGRU	Word2Vec	0.9294	0.9246	0.9420	0.9332
CNN-LSTM	Word2Vec	0.9290	0.9291	0.9357	0.9324
CNN-GRU	Word2Vec	0.9332	0.9342	0.9384	0.9363
CNN-BiLSTM	Word2Vec	0.9304	0.9362	0.9304	0.9333
CNN-BiGRU	Word2Vec	0.9070	0.9656	0.8527	0.9056
Hybrid	Word2Vec	0.9294	0.9145	0.9545	0.9340
CNN	GloVe	0.9542	0.9554	0.9571	0.9563
BiLSTM	GloVe	0.9355	0.9161	0.9652	0.9400
BiGRU	GloVe	0.9421	0.9220	0.9714	0.9461
CNN-LSTM	GloVe	0.8593	0.7991	0.9768	0.8791
CNN-GRU	GloVe	0.5234	0.5234	0.9991	0.6869
CNN-BiLSTM	GloVe	0.9477	0.9308	0.9723	0.9511
CNN-BiGRU	GloVe	0.9472	0.9382	0.9625	0.9502
Hybrid	GloVe	0.9374	0.9143	0.9714	0.9420
CNN	BERT _{base}	0.9752	0.9725	0.9804	0.9764
BiLSTM	BERT _{base}	0.9650	0.9485	0.9866	0.9672
BiGRU	BERT _{base}	0.9706	0.9616	0.9830	0.9722
CNN-LSTM	BERT _{base}	0.9598	0.9434	0.9821	0.9624
CNN-GRU	BERT _{base}	0.9626	0.9710	0.9571	0.9640
CNN-BiLSTM	BERT _{base}	0.9621	0.9529	0.9759	0.9643
CNN-BiGRU	BERT _{base}	0.9565	0.9485	0.9696	0.9589
Hybrid	BERT _{base}	0.9617	0.9642	0.9625	0.9634
BERT _{base}	BERT _{base}	0.9771	0.9735	0.9830	0.9782
RoBERTa _{base}	RoBERTa _{base}	0.9668	0.9541	0.9839	0.9688

Figure 12: Fake news detection using advanced ML and DL classification models on COVID-19 dataset. [3]

Table 5 Overall model performance

Methods	Dataset					
	Pymedia			Politifact		
	Accuracy	F1 score	AUC	Accuracy	F1 score	AUC
3HAN	0.60	0.76	0.58	0.32	0.47	0.50
HNN	0.62	0.75	0.49	0.52	0.28	0.49
CNN-RNN	0.77	0.84	0.87	0.71	0.56	0.77
CNN-LSTM	0.75	0.83	0.78	0.66	0.42	0.76
BERT-NLI	0.73	0.46	0.78	0.75	0.79	0.83
FakeBERT	0.69	0.81	0.54	0.73	0.34	0.60
Proposed Approach	0.83	0.74	0.88	0.84	0.87	0.93

We surpass all the state-of-the-art methods

Figure 13: Overall Model performance [6]


```
real_news.head()
```

	title	text	subject	date
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017


```
real_news.tail()
```

	title	text	subject	date
21412	'Fully committed' NATO backs new U.S. approach...	BRUSSELS (Reuters) - NATO allies on Tuesday we...	worldnews	August 22, 2017
21413	LexisNexis withdrew two products from Chinese ...	LONDON (Reuters) - LexisNexis, a provider of L...	worldnews	August 22, 2017
21414	Minsk cultural hub becomes haven from authorities	MINSK (Reuters) - In the shadow of disused Sov...	worldnews	August 22, 2017
21415	Vatican upbeat on possibility of Pope Francis ...	MOSCOW (Reuters) - Vatican Secretary of State ...	worldnews	August 22, 2017
21416	Indonesia to buy \$1.14 billion worth of Russia...	JAKARTA (Reuters) - Indonesia will buy 11 Sukh...	worldnews	August 22, 2017

Figure 14: Language pattern in real news dataset before removal

```
# Remove everthing before the hyphen using regular expression
real_news['text'] = real_news['text'].str.replace(r'^.*?-', '', regex=True)
```



```
real_news.head()
```

	title	text	subject	date
0	As U.S. budget fight looms, Republicans flip t...	The head of a conservative Republican faction...	politicsNews	December 31, 2017
1	U.S. military to accept transgender recruits o...	Transgender people will be allowed for the fi...	politicsNews	December 29, 2017
2	Senior U.S. Republican senator: 'Let Mr. Muell...	The special counsel investigation of links be...	politicsNews	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	Trump campaign adviser George Papadopoulos to...	politicsNews	December 30, 2017
4	Trump wants Postal Service to charge 'much mor...	President Donald Trump called on the U.S. Pos...	politicsNews	December 29, 2017

Figure 15: Removal of pattern in text column using regex

```
Epoch Batch: 1 / 3
Training in progress, please wait...

Average training loss value: 0.67
Time taken for training epoch : 1:04:33

Running tests on Model Validation...
Validation Loss: 0.67
Validation took: 0:08:32

Epoch Batch: 2 / 3
Training in progress, please wait...

Average training loss value: 0.62
Time taken for training epoch : 1:10:09

Running tests on Model Validation...
Validation Loss: 0.66
Validation took: 0:07:36

Epoch Batch: 3 / 3
Training in progress, please wait...

Average training loss value: 0.60
...

Running tests on Model Validation...
Validation Loss: 0.65
Validation took: 0:08:04
```

Figure 16: Recording of values during training process

```
import torch
from transformers import BertForSequenceClassification, BertTokenizer
import pickle
# Load BERT model and tokenizer
model = BertForSequenceClassification.from_pretrained('bert-base-uncased')
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Save the model and tokenizer
model_path = './model/bert_model.pkl'
tokenizer_path = './model/tokenizer.pkl'

# Save the model using torch.save()
torch.save(model.state_dict(), model_path)

# Save the tokenizer using pickle
with open(tokenizer_path, 'wb') as f:
    pickle.dump(tokenizer, f)
```

Figure 17: Code used when saving the current BERT model

```

app = Flask(__name__)
CORS(app) # Enable CORS for all routes

# Load the BERT model and tokenizer
model_path = "bert-base-uncased"
tokenizer = BertTokenizer.from_pretrained(model_path)
model = BertForSequenceClassification.from_pretrained(model_path)
model.eval()

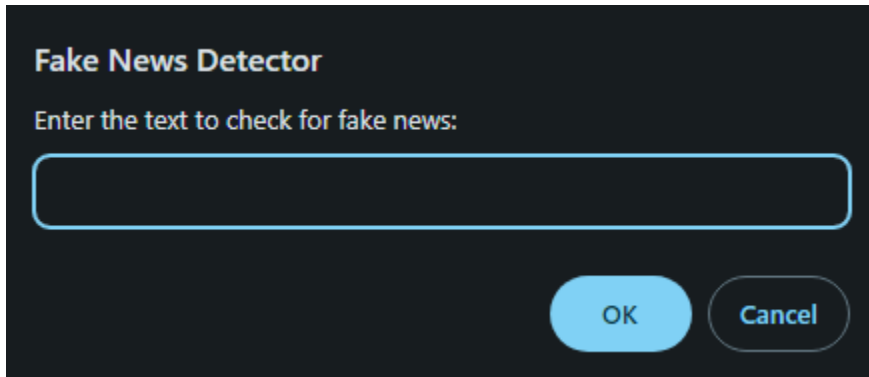
# Define function for making predictions
def predict(text):
    inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True)
    with torch.no_grad():
        outputs = model(**inputs)
        predicted_class = torch.argmax(outputs.logits).item()
    return predicted_class

@app.route('/predict', methods=['POST'])
def predict_route():
    data = request.json
    text = data['text']
    predicted_class = predict(text)
    return jsonify({"predicted_class": predicted_class})

if __name__ == '__main__':
    app.run(debug=True)

```

Figure 18: Code for Flask server



Fake News Detector

Enter the text to check for fake news:

OK Cancel

Figure 19: Text input box in Chrome extension

```

extension > JS content_script.js > ...
1  // Function to extract text from the webpage
2  function extractText() {
3      return document.body.innerText;
4  }
5
6  // Function to send text to Flask server for prediction
7  function sendTextForPrediction(text) {
8      fetch('http://localhost:5000/predict', {
9          method: 'POST',
10         headers: {
11             'Content-Type': 'application/json',
12         },
13         body: JSON.stringify({ text: text }),
14     })
15     .then(response => response.json())
16     .then(data => {
17         console.log('Predicted Class:', data.predicted_class);
18         alert('Predicted Class: ' + (data.predicted_class ? 'Safe' : 'Not Safe'));
19     })
20     .catch(error => {
21         console.error('Error:', error);
22     });
23 }
24
25 // Extract text from the webpage and send it for prediction
26 const text = extractText();
27 sendTextForPrediction(text);

```

Figure 20: /predict endpoint in Flask server

```

extension > JS popup.js > ...
1  document.addEventListener('DOMContentLoaded', function() {
2      const checkButton = document.getElementById('checkButton');
3
4      checkButton.addEventListener('click', function() {
5          const text = prompt("Enter the text to check for fake news:");
6          if (text !== null && text.trim() !== '') {
7              fetch('http://localhost:5000/predict', {
8                  method: 'POST',
9                  headers: {
10                      'Content-Type': 'application/json',
11                  },
12                  body: JSON.stringify({ text: text }),
13              })
14              .then(response => response.json())
15              .then(data => {
16                  const message = data.predicted_class === 1 ? "The text you have input appears to be safe" : "The text you have input may contain fake news";
17                  alert(message);
18              })
19              .catch(error => {
20                  console.error('Error:', error);
21              });
22          }
23      });
24  });

```

Figure 21: Function for predicting class

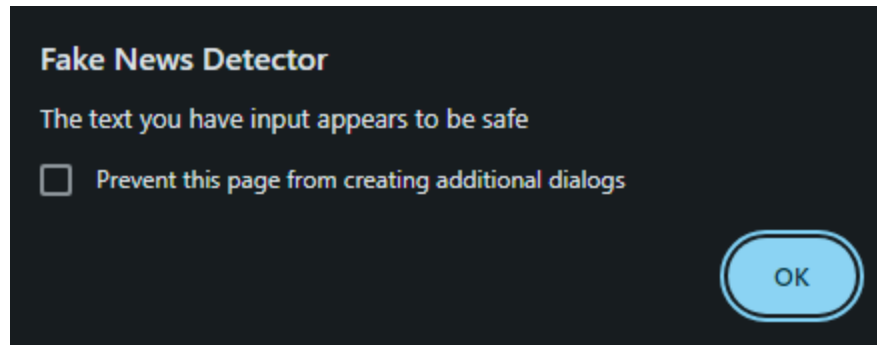


Figure 22: Alert Message after inputting text into extension

The Google Chrome extension is easy to add to my browser.

17 responses

 Copy

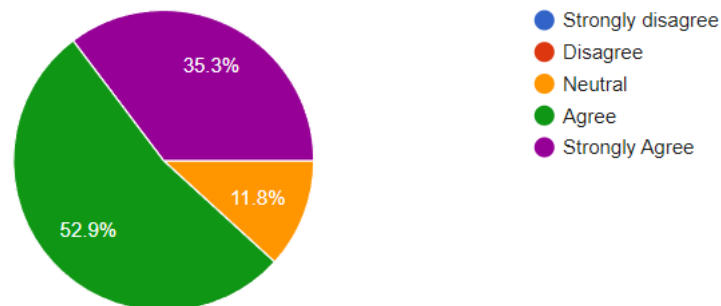


Figure 23: Question on ease of adding the extension to the browser

The Google Chrome extension allows me to easily send pieces of text from news articles I come across to validate.

17 responses

 Copy

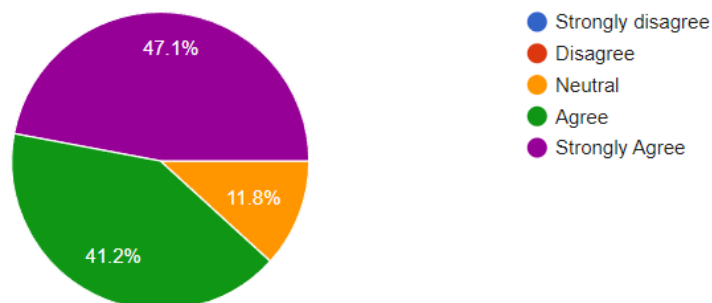


Figure 24: Question on ease of validating text

Personally, I would use this or similar extensions to validate my news sources in the future, given further improvements.

 Copy

17 responses

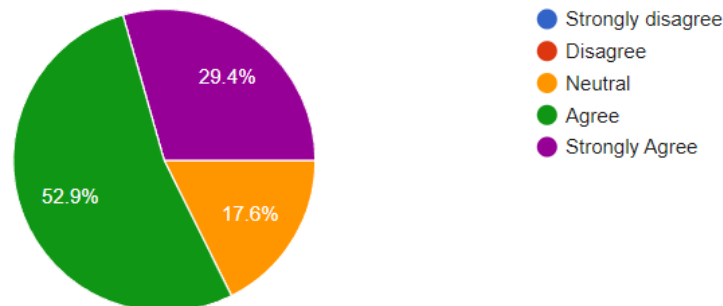


Figure 25: Question on whether users would use variations of fake news extensions

Suggest any improvements to the Google Chrome extension that would improve its usability, as well as any other areas of improvement.

5 responses

Instead of having to initiate a python flask server. If the BERT model can be housed within the chrome extension itself, it would be easier for non IT expert users to use the chrome extension.

In the future, the popup will automatically appear for the main news title that appears on the webpage

Implement a function to automatically validate all the text displayed on the webpage and inform the user based on the output from the model after analysing the whole page

Remove the need to start a python server in case the user does not have python installed

Add more buttons to use different models for text checking

Figure 26: Google Form Feedback for potential improvements for Chrome Extension