FACTORIAL!: A GOOGLE CHROME EXTENSION TO ANALYZE AND RATE NEWS ARTICLES USING MACHINE LEARNING

Daniel Miao[1], Tyler Liu[2], Andrew Park[3]

[1]The Harker School, 500 Saratoga Ave., San Jose, CA 95129
[2]The Webb Schools, W. Baseline Road, Claremont, CA 91711
[3]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

The advent and spread of the internet has caused many users to favor the convenience and breadth of reporting that online news offers, whether it be from media companies or social platforms, which in turn has led to the monetization and corruption of said stories [4]. Large, company-owned news sites each try to appeal to only a few groups across the political spectrum, oftentimes sacrificing the indifference and integrity which serve as the tenets of honest journalism. We propose to aid in solving this problem a Chrome extension which serves to provide metrics, information, and line-by-line analysis of article text in order to help readers stay aware and healthily skeptical [5][6]. Using machine learning (ML) as well as traditional algorithms, we aim to provide key info on the article's truthfulness as well as the source's bias and ownership [7]. In this project, we used 3 main models, each to detect fake news, political leaning and sentiment, in addition to traditional criteria such as readability, # of words, and time to read. All of our models performed well both theoretically and practically, giving above 80% accuracy on all occasions.

KEYWORDS

Artificial Intelligence, Fake News, Chrome Extension, Text Analysis

1. INTRODUCTION

The loss of small, local print newspapers in favor of online mass media owned by large companies has led to the widespread monetization and monopolization of news [8]. This, in turn, has favored eye-catching titles and dramatic writing over truthful reporting aiming to represent the current state of the world accurately. Additionally, partisan politics in the USA and the polarization of the general populace have incentivized news companies to appeal to specific, target audiences in order to maximize profits; this may take the form of subtle vilification of political rivals through wordplay, blatant misrepresentation of facts and quotes, treating hypotheses as if they were facts, or even the omission of certain topics entirely. Even if one recognizes these journalistic fallacies and actively tries to avoid them, the consolidation of news by a small number of large companies has made it especially difficult.

The advent of online interconnectedness has exacerbated the control that news companies possess over the average citizen. 86% of Americans receive the majority of their information on the world around them through the internet, most notably social media and news [9]. In the long term, subconscious nudges by the internet radicalize voters and lead to more and more extreme

David C. Wyld et al. (Eds): NSEC, InWeS, CSIP, BMLI, CACIT, EDUIT -2023
pp. 39-50, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.131504

policies, encouraging civil unrest and violence among those who would previously have never considered it. As of right now, misinformation perpetrated by private individuals does not receive punishment nor recognition, and online echo chambers quickly shut down attempts to bring more centric ideas into their feeds.

It becomes especially difficult to fight back against this new wave when you consider that large media companies are behind this, all the while being the ones to control the information the average consumer sees day-to-day. They have repeatedly shown that they are not below insults and other forms of vituperation when it comes to their political rivals or anyone seeking to shift the climate away from the fringes of the political spectrum and back toward the center [10]. With these, it is apparent that effectively reaching the consumer is extremely difficult.

The other literature we examined in the methodology section each tried to combat the spread of fake news and misinformation through the use of artificial intelligence – however, where each differed was the type of model they chose to utilize in order to accomplish this task and what kind of application they chose to use. The first used a Logistic Regression model and put it into practicality through a website, the second used a naive Bayes classifier, and the third used a Facebook Messenger Chatbot in order to spread its usage to a wider audience [11]. Our project improves on these partially in accuracy but majorly in the application we choose to present it in. Along with a simple fake news AI, we give a much larger amount of metrics and use in-article phrase warnings in order to present to the reader a much more critical and skeptical mindset [12]. Additionally, the form function of a Chrome extension is much more alluring than other methods, having improved reliability and usability towards a wider audience.

Our proposal to address the ever-growing amount of misinformation is to provide a comprehensive extension service providing readers with tools and context to help them stay aware and critical of the information they are consuming. We aim to take advantage of the tools provided by the digital age to automate much of what would previously be tedious work; our extension will be split into 3 parts:

1. Front-end: We will integrate the extension directly into the web pages that the user browses, allowing for a seamless, aesthetically friendly UI which will not serve to push away potential users. Above articles themselves we will provide a block of information detailing the results of the numerous AI tests we will conduct on the article title and content itself, and also standard metrics such as readability, reading time, number of quotes, and also the number and magnitude of possibly misleading words or phrases. These will be explained later in more detail.

2. Source Checking: Through non-profit, open-source websites such as Political Personality and AllSides, we will collect a range of data and provide them in an easy-to-understand form through the use of a fixed, collapsible blob in the bottom right of the viewport. These metrics will include: the general rating of the company

3. Primitive Fact-Checking: We aim to utilize the newest features of ChatGPT provided by OpenAI, most notably the feature to inject context and internet data into the model, in order to attempt primitive automatic fact checking [13]. When a user highlights a block of text, it will prompt the user if they wish to check the text that has been highlighted, and if they select “yes,” it will provide them with whether the fact is true, false, or if the AI is unsure due to vague wording or an incomplete claim. It will also attempt to provide relevant sources and quotes from within those sources.

The two major experiments we conducted were meant to expose political biases within the models used to provide more abstractive measurements on the article's contents. Following two of our metrics, Fake News and Political Lean, we conducted two experiments, one for each. The former aimed to determine whether or not the AI had any training biases that made it more likely to miscategorize one political lean as another; using confusion matrices and analyzing the data revealed that the AI was more likely to miscategorize left-leaning articles, and that left-center was the most common bias to be misattributed. The latter also wanted to check political lean but for the fake news model; for this, we attached bias labels to the training data based on their respective source and checked to see its accuracy with all three leanings: left, right, and center. The results revealed that the skew of the dataset favored marking left-leaning articles as true and right-leaning articles as fake.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. A Lack of Relevant Data

A lack of relevant data regarding the metrics we wish to address may lead to incorrect results from the models trained on said data. Thus, it would be unwise to rely on only a few models; in order to solve this problem, we should increase the number of metrics and thus models, thus reducing the chance of error. We could also combine datasets in order to form more rounded data, thus producing AIs capable of accuracy in a greater number of scenarios

2.2. Server Hosting Method

In order for a seamless and consistent user experience, we must choose the correct server hosting method for our Chrome Extension. Initially, for testing we used our own home computers to host the servers on localhost; however, once we sought to expand the reach of our program to others it was clear that another solution was imperative. Hosting on personal computers was simply too unreliable as the uptime of the server was dependent on the decisions of the hoster, and unforeseen consequences could arise during power outages or simply from accidents or malfunctions. To solve this, we could use an online server hosting website – for our application we nominated repl.it as it was simple, inexpensive, and reliable.

2.3. Privacy and Security

In order to help our users understand their own political biases, our application collects data regarding the articles the user visits and documents those in the user's profile. One issue is that of privacy and security – the information we collect is very sensitive and needs to be properly protected. As a small project, we do not currently possess the funds to adequately protect our data alone. As thus, we could use a combination of locally storing data and using services from larger companies such as Firebase.

3. SOLUTION

Our project will have 3 major components: the extension, the server, and the firebase authentication database. The extension, being the vehicle through which the user interacts with the other components, always comes first. It handles the web scraping and preprocessing of data, along with the UI and the design features. The server receives said data, and runs the models stored within to gather metrics and criterion on the data the extension has sent, which it will then

give back to the extension and thus the user. The firebase database allows for user authentication and the storing of user data, which allows for features such as the tracking of the political mix the user normally interacts with.

When the user first loads a news article, the extension will recognize this and immediately begin scraping data from the webpage: the title and the article. It will then send these to the server, which will run these through 4 major models and send the results back to the extension. Once the extension has gotten these results, it will begin loading the UI: the source blob, the article blob, and the underline blob. Each has a purpose: the source blob tells the user info about the source: who owns it, where it operates, its political lean, etc. The article blob contains the 4 metrics the extension receives from the server as well as additional metrics calculated by the article: readability, # of quotes, etc. The underline blob scans the article text for certain phrases that are often used to mislead or misrepresent, and underlines and highlights these with colors corresponding to their threat levels, adding descriptive boxes underneath when highlighted. Finally, when these three blobs are loaded, the extension takes the political lean of the article/source and sends it to the firebase database, which will then update the user's profile based on it.

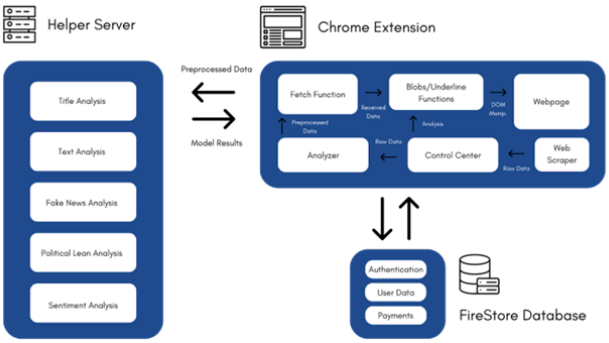


Figure 1. Overview of the solution

The first major component of Factorial is the Flask server itself. The Flask server has several different responsibilities including both analysis of data using trained AI models and hosting helper functions that extension clients might need. The server essentially waits for clients to send analysis requests where it takes the data given by them and runs them through the stored AI models before then sending back the result for the extension to use. As far as helper functions go, clients can send the server requests to perform certain tasks that they cannot do on their own.

```python
@app.route('/api/metrics', methods=['POST'])
def get_metrics():

  article = request.get_json()

  title = article['Title']
  content = article['Content']

  metrics = f.get_metrics({'Title': title, 'Content': content})

  return jsonify(metrics)


@app.route('/api/transcripts', methods=['POST'])
def get_transcripts():

  data = request.get_json()

  video_id = data['videoId']

  if not video_id:
    return jsonify({'error': 'No video ID provided'}), 400

  try:
    transcript = YouTubeTranscriptApi.get_transcript(video_id)
    print(transcript)
  except Exception as e:
    return jsonify({'error': str(e)}), 500

  return jsonify(transcript)
```

Figure 2. Screenshot of code 1

```python
def get_metrics(article):

    t_title = tokenize(article['Title'])
    t_content = tokenize(article['Content'])
    # article_authors_info = get_author_string(article['Authors'])

    metrics = {'tfn': int(predict('title', t_title)),
               'tcb': int(clickbait(article)),
               'tpl': int(predict('reddit', t_title)),
               # 'aai': article_authors_info,
               'cfn': int(predict('text', t_content)),
               'csm': int(predict('sentiment', t_content))
               }

    return metrics
```

Figure 3. Screenshot of code 2

The two app routes that are shown here from the Flask server showcase each of the primary responsibilities of Factorial to deliver information to the end user [15]. The first get_metrics function takes the content from the extension that called it via POST request and passes it on to its stored AI models. The models are invoked in the predict function which dictates what model should be used to get relevant information. The second app route is one that extensions can use to request the transcript of a youtube video. The reason we had to do this is that extensions are no longer able to use Youtube's API directly to get transcripts for security reasons and so we instead utilize a Python tool which allows us to overcome the limitations of the extension itself. Functions like this are created within the Flask server to help unlock the capabilities of the client beyond what it may normally be able to do otherwise.

Another core component of Factorial is the browser extension. This is the client-side part of the system that directly interacts with the user. We chose to make the frontend part an extension as it is more convenient for the end user to get our analysis right away alongside the article they are reading. This also allows the extension to directly access the content the user is looking at so that it can package and preprocess that information to send to the server.

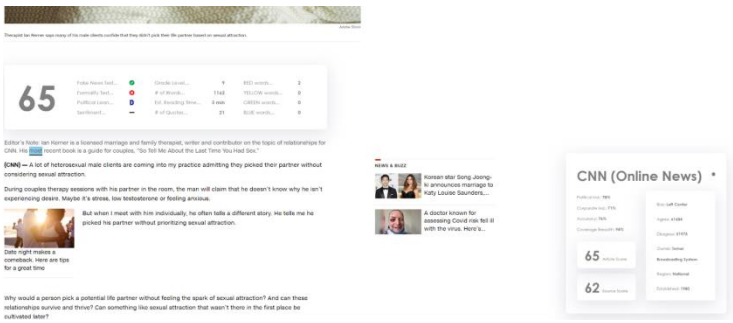


Figure 4. Screenshot of website

```
async function fetchData() {
  try {

    var requestData = {
        Title: titleString,
        Content: contentString,
    };

    const metrics = await fetch('https://david-and-tyler-hosting-3.youngbuck9696.repl.co/api/metrics', {
        method: 'POST',
        mode: 'cors',
        headers: {"Content-Type": "application/json"},
        body: JSON.stringify(requestData),
    })

    const res = await metrics.json();
    data = res;

  } catch (error) {
    console.error(error);
  };
}

async function fetchSourceData()
{
  try
  {
    const ppResponse = await fetch(chrome.runtime.getURL('blobs/data/PoliticalPersonality.json'));
    const ppData = await ppResponse.json();

    const assResponse = await fetch(chrome.runtime.getURL('blobs/data/Allsides.json'));
    const assData = await assResponse.json();

    const ppJSONstring = JSON.stringify(ppData);
    const assJSONstring = JSON.stringify(assData);

    const ppDataDict = JSON.parse(ppJSONstring);
    const assDataDict = JSON.parse(assJSONstring);

    const hostkey = "https://" + hostname + "/";

    let pp, ass;

    if (hostkey in ppDataDict) {
      pp = ppDataDict[hostkey];
    } else{
      pp = {
        'Political Personality Rating': '-1',
        'Political Independence': 'NaN',
        'Corporate Independence': 'NaN',
        'Accuracy': 'NaN',
        'Coverage Breadth': 'NaN'
      };
    }

    if (hostkey in assDataDict) {
      ass = assDataDict[hostkey];
    } else{
      ass = {
        'News Source': 'NaN',
        'Allsides Bias Rating': 'NaN',
        'Agree': 'NaN',
        'Disagree': 'NaN',
        'Owner (Text)': 'NaN',
        'Region (Text)': 'NaN',
        'Established (Text)': 'NaN'
      };
    }

    return [pp, ass];
  }
  catch (error)
  {
    console.error(error);
  }
}
```

Figure 5. Screenshot of code 3

The code shown above details the functions which the extension uses to procure the data necessary for the source and underline blobs. In the first snippet, the extension calls a fetch on its local files, converting the JSON into readable javascript objects, from which it can be parsed based on the webpage the under is currently on. In the second, the extension takes the article content and scans it for signal phrases (taken from another JSON file), calculating readability, total # of words, and # of quotes along the way. For readability, we use the Flesch-Kincaid method, one of the most commonly used forms of readability measurement. Once the required metrics have been calculated, the extension then creates blobs underneath each word to explain to the user why they may be potentially harmful.

The Firebase component is the part that is responsible for both user authentication and cloud-based account data. A notable benefit contributing to our decision to use Firebase is that it is a scalable BaaS (backend-as-a-service) which means that we will only ever incur costs if there is enough usage by users to require it. In order to comply with the security measures put in place from Manifest V3, we had to include a localized copy of the Firebase library and interface with it via a service worker so that the extension would be able to interact with the rest of the project.

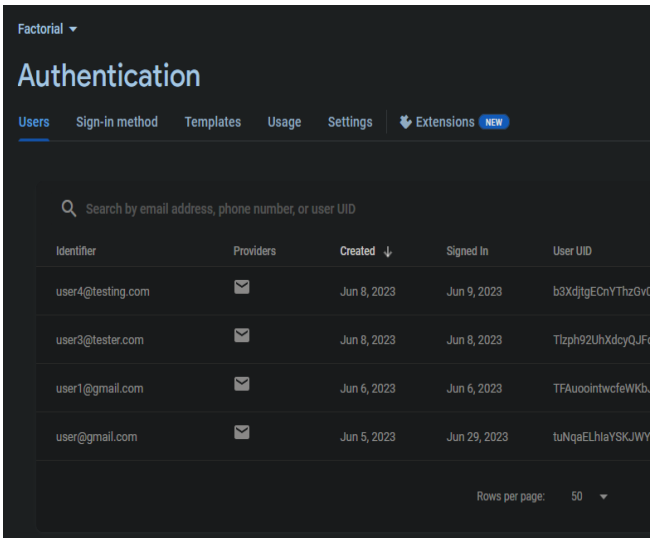


Figure 6. Screenshot of authentication

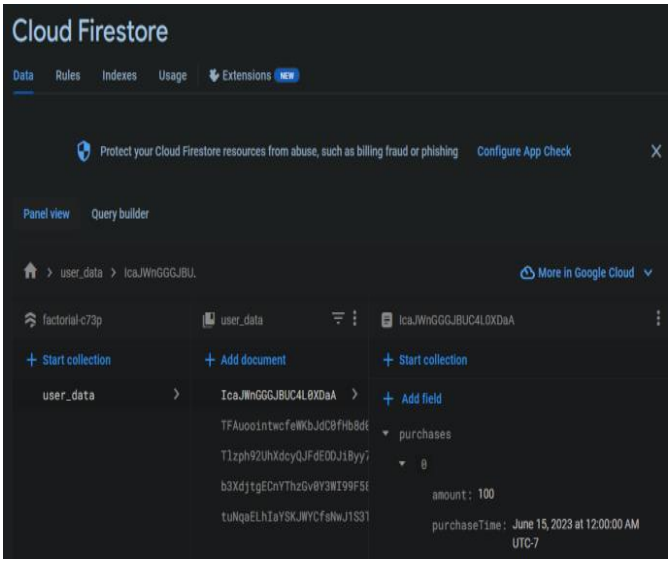


Figure 7. Screenshot of cloud firestore

```
firebase.initializeApp(firebaseConfig);
console.log("Firebase is initialized");

chrome.runtime.onMessage.addListener((msg, sender, resp) => {
    if (msg.command == "signup") {
        const db = firebase.firestore();
        console.log("Attempting to create user with email: " + msg.email + " and password: " + msg.password + ".");
        firebase.auth().createUserWithEmailAndPassword(msg.email, msg.password).then((value) => {
            console.log("User successfully created.");
            db.collection("user_data").doc(value.user.uid).set(data);
            resp({result: "success"});
        },
        ).catch(function(error) {
            console.log("User creation failed.");
            resp({result: "failure"});
        });
        return true;
    }

    if (msg.command == "login") {
        console.log("Attempting to login user with email: " + msg.email + " and password: " + msg.password + ".");
        firebase.auth().signInWithEmailAndPassword(msg.email, msg.password).then(function() {
            console.log("User successfully logged in.");
            resp({result: "success"});
        }).catch(function(error) {
            console.log("User login failed.");
            resp({result: "failure"});
        });
        return true;
    }

    if (msg.command == "logout") {
        console.log("Attempting to logout user.");
        firebase.auth().signOut().then(function() {
            console.log("User successfully logged out.");
            resp({result: "success"});
        }).catch(function(error) {
            console.log("User logout failed.");
            resp({result: "failure"});
        });
        return true;
    }

    if (msg.command == "checkAuth") {
        var user = firebase.auth().currentUser;
        if (user) {
            console.log("User is logged in.");
            resp({result: "loggedIn"});
        } else {
            console.log("User is not logged in.");
            resp({result: "loggedOut"});
        }
        return true;
    }

    if (msg.command == 'getCreationDate') {
        const user = firebase.auth().currentUser;
        if (user) {
          // Format the creation timestamp
          const creationDate = new Date(user.metadata.creationTime);
          const formattedDate = creationDate.toDateString();

          // Send the creation date back to the sender
          resp({ creationDate: formattedDate });
        }
      }

    if (msg.command == 'getUserData') {
```

Figure 8. Screenshot of code 4

This segment of code is from the service worker we set up in the extension to allow it to work with Firebase. We start off by initializing Firebase before then setting up a listener. This listener corresponds to the extension's sign-in and sign-out pages, where several buttons are kept. Each button has an ID which corresponds to an if statement within this code. When a button is pressed, its corresponding listener goes off and performs the action necessary through a connection with Firebase. For example, when the button is clicked to sign up, the extension takes the text from within the two input boxes (username and password) and uses the function firebase.auth().createUserWithEmailAndPassword() to create a user from said data.

4. EXPERIMENT

4.1. Experiment 1

The AI may provide accuracy biases stemming from its training data – it may incorrectly categorize one bias more than another, leading to a skew in its direction. For factors such as political lean and fake news, this can be particularly misleading to the viewer.

The most effective way of gauging an AI's bias is with a confusion matrix. Confusion matrices compare the AI's output with the correct result and illustrate this data in a simplistic, easy-to-understand table, with shades of coloring representing the most accurate or least accurate portions of the AI's understanding. However, for this to work the data must also be accurate. We will source the data from an extremely large dataset and tag the data based on the general metrics of the source itself, without revealing this to the model. Although there will be exceptions, in general the sheer amount of data will cause the correct labels to align with the general trends, and the AI's weakest points will show through regardless of these outliers.

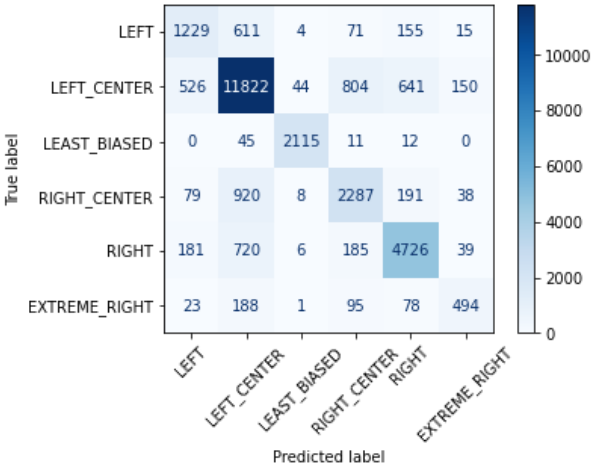


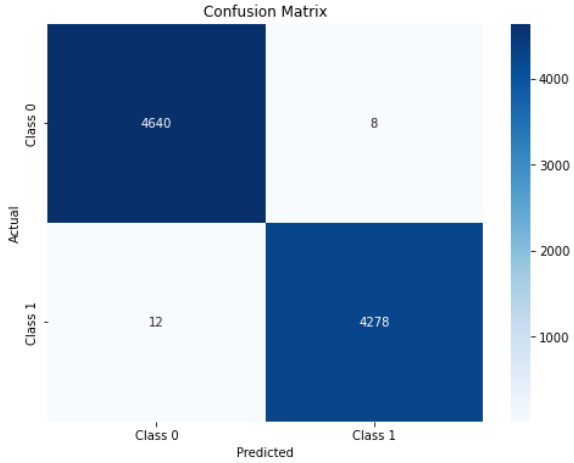
Figure 9. Figure of experiment 1

The general accuracy of the AI (comparing the cumulative sum of the correct predictions and that of the incorrect ones) is 80%. Within other literature other researchers obtained similar results, so this looks satisfactory. At first viewing, it seems that the AI is most accurate with the left-center bias, but this may be caused by a prevalence of that category within the training data. Likewise, the least accurate is that of the extreme right, however we must also take this result with the same considerations as the prior. It seems the left side is a bit less accurate than the right side (mainly from miscategorizations of left-center), perhaps from a lack of repeated words/phrases/sentence structure that the other side shares. Taking into account the outliers, perhaps the amount of incorrect predictions is lower or higher than what we see, but as I mentioned before the most probable outcome is that the false positives and the false negatives balance each other out given the large size of the training data.

4.2. Experiment 2

Another potential blind spot may be the fake news detection AI – fake news can be extremely subtle and the AI may miscategorize easily, especially if there was too little training data or if said data was not representative of a wide range of fake news. The main concern is that the AI may mark one side as fake news more often than another.

Unlike in the previous experiment, we are not able to simply run a confusion matrix on the model to determine its bias – instead, we can take a look at the training data and get how many articles of left-leaning, center, or right-leaning sources are labeled as fake or non-fake news, also checking how well the AI's model aligns with this data (represented by the model's accuracy). Following this, we can take the same approach and apply it to the AI themselves, giving it verified articles from a range of different sources and seeing which side it more often marks as false.




```
[60] t_value_counts = true['lean'].value_counts()
     t_value_counts

     1    8696
     4    6893
     3    2103
     2    1947
     5    1128
     0     650
     Name: lean, dtype: int64

     f_value_counts = fake['lean'].value_counts()
     f_value_counts

     4    12390
     1     5871
     0     3119
     3     1229
     5      831
     2       41
     Name: lean, dtype: int64
```

```
t_percentages = (t_value_counts / len(true['lean'])) * 100
t_percentages

1    40.603259
4    32.184713
3     9.819302
2     9.090909
5     5.266844
0     3.034972
Name: lean, dtype: float64

f_percentages = (f_value_counts / len(fake['lean'])) * 100
f_percentages

4    52.766066
1    25.003194
0    13.283080
3     5.234019
5     3.539032
2     0.174609
Name: lean, dtype: float64
```

Figure 10. Figure of experiment 2

The training data was mostly filled with left-leaning and right-center-leaning articles, which, given the model's accuracy of 99.8%, indicates that it should have a tangible amount of bias toward those sides. The third image shows that the "true" articles were populated with 40% articles from the left and 32% from the right center, and for the fake articles there was 52% for the right-center as opposed to only 25% from the left (the scale goes from 1-5 and left-right

respectively). Extrapolating, the data suggests that in general the model may have a bias towards marking left-leaning articles as true and right-leaning ones as false; however, this may not be completely unsubstantiated. The AI's bias may serve as a blind spot for practical applications and thus prompt future remedies, perhaps through more equalized data or simply larger datasets.

5. RELATED WORK

In their paper "Fake News Detection using Machine Learning Algorithms" by Uma Sharma et al., they use scikit-learn, Django, and a logistic regression model in order to attempt binary classification of fake news [1]. Their final model produces an accuracy of 65%, however with grid search parameter optimization it is increased to 80%. Finally, they put their model into use through a website from which one can type a headline and have a result returned. Their approach mirrors that of ours, however our content model has a higher accuracy level on our data. Additionally, instead of looking at only the headline, we have 2 models, one which checks the title of the model and another which checks the text of the model, in order to separate clickbait and rage-baiting from real fake news. Our model is also packaged into a more user-friendly application, one which automatically runs whenever it is on a news website.

In their paper "Fake news detection using naive Bayes classifier," Mykhailo Granik et al. attempts to use the titular classifier in order to detect fake news within a large dataset of Facebook posts [2]. Their final model possesses an accuracy of 74%, which they claim is satisfactory considering the complexity, or rather its absence, of the model itself. In their paper, they describe the major limitation as being the skew towards real news (95.1% real as opposed to 4.9% fake) which they claim as the major driving factor behind its lower accuracy for fake news. Based on the dataset we used, we achieved an accuracy of 99.8%, which is higher than their results. Additionally, we package our multiple models into an easy-to-use chrome extension, which brings practical applications as well as theoretical ones.

In their paper "Automatic Online Fake News Detection Combining Content and Social Signals," Marco L. Della Vedova et al. attempt to use machine learning to combine content analysis and social context methodologies into one single Facebook Messenger application, outperforming other models in the literature by 4.8% for an accuracy of 78.8% [3]. Our model's accuracy is quite similar to theirs, however where we attempt to improve is the incorporation of these models and much more into a Google chrome extension, where it can be used by the vast majority of Chromium users rather than simply those who browse Facebook.

6. CONCLUSIONS

Limitations arise both on a small scale and a larger scale. On a larger scale, when you take into consideration factors such as the omission of certain topics which challenge or contradict the message the news company is trying to convey, this extension cannot directly address this. For this, a web application which gathers all of the articles from multiple sources using web scraping and uses generative AI to take and condense the information contained within the articles may be more effective, however that is a project for the future. Within the range of features this project is able to cover, there are 2 main limitations: one regarding the accuracy, or rather accuracy, inherent to artificial intelligence models and the other pertaining to the sheer number of ways an article can mislead or misrepresent. Factorial! Cannot solve them all, and that is why it is to be taken as a supplement to daily reading and not a complete package to rely on.

In conclusion, this project attempts to promote critical thinking and healthy skepticism when reading news articles through the use of AI-generated metrics, source analysis, and the marking

of specific signal phrases commonly used to mislead, all gathered within the accessibility and simplicity of a Google Chrome extension [14].

REFERENCES

[1] Sharma, Uma, Sidarth Saran, and Shankar M. Patil. "Fake news detection using machine learning algorithms." International Journal of Creative Research Thoughts (IJCRT) 8.6 (2020): 509-518.
[2] Granik, Mykhailo, and Volodymyr Mesyura. "Fake news detection using naive Bayes classifier." 2017 IEEE first Ukraine conference on electrical and computer engineering (UKRCON). IEEE, 2017.
[3] Della Vedova, Marco L., et al. "Automatic online fake news detection combining content and social signals." 2018 22nd conference of open innovations association (FRUCT). IEEE, 2018.
[4] Arendt, Rosalie, et al. "Comparison of different monetization methods in LCA: A review." Sustainability 12.24 (2020): 10493.
[5] Carlini, Nicholas, Adrienne Porter Felt, and David Wagner. "An evaluation of the google chrome extension security architecture." 21st USENIX Security Symposium (USENIX Security 12). 2012.
[6] Reinking, Johannes, et al. "First high-resolution spectrum and line-by-line analysis of the 2ν2 band of HTO around 3.8 microns." Journal of Quantitative Spectroscopy and Radiative Transfer 230 (2019): 61-64.
[7] Jordan, Michael I., and Tom M. Mitchell. "Machine learning: Trends, perspectives, and prospects." Science 349.6245 (2015): 255-260.
[8] Lesicko, Christine Katherine. "The New News: Challenges of Monetization, Engagement, and Protection of News Organizations' Online Content." Reynolds Ct. & Media LJ 2 (2012): 339.
[9] Shearer, Elisa. "More than eight-in-ten Americans get news from digital devices." Pew Research Center 12 (2021).
[10] Crawford, Jarret T., et al. "Social and economic ideologies differentially predict prejudice across the political spectrum, but social issues are most divisive." Journal of personality and social psychology 112.3 (2017): 383.
[11] Balasudarsun, N. L., M. Sathish, and K. Gowtham. "Optimal ways for companies to use Facebook Messenger Chatbot as a Marketing Communication Channel." Asian Journal of Business Research 8.2 (2018): 1-17.
[12] Fetzer, James H., and James H. Fetzer. What is Artificial Intelligence?. Springer Netherlands, 1990.
[13] Surameery, Nigar M. Shafiq, and Mohammed Y. Shakor. "Use chat gpt to solve programming bugs." International Journal of Information Technology & Computer Engineering (IJITC) ISSN: 2455-5290 3.01 (2023): 17-22.
[14] Chen, Junyu, et al. "Learning to Evaluate the Artness of AI-generated Images." arXiv preprint arXiv:2305.04923 (2023).
[15] Walsh, Eamon. "Application of the flask architecture to the x window system server." Proceedings of the 2007 SELinux Symposium. 2007.