Fake News Detector

Internet Browser Extension

Cloud Computing Systems

January 4th, 2022

Supervisors:

- Aneta Poniszewska-Marańda, DSc, PhD, MSc
- Lumbardha Hasimi, MSc

Team:

- Eliza Lech, 230455
- Adam Czyżewski, 230447
- Karol Kowalski, 230452
- Artur Miłosz, 230457

Table of contents

1. Abstract

Articles containing fake news are wide spreading and are a matter of serious concern because of their possibly damaging effect on societies. They result in disinformation and media chaos. Therefore, currently much research is carried out in order to find a way of detecting fake news. This paper presents results of work on an extension for Internet browsers, the aim of which is to make a prediction on falsity of the article opened by the user. The whole system consists in a browser extension designed with JavaScript environment, and a cloud function configured on Google Cloud platform, where a Neural Network model based on TensorFlow python library makes predictions on the articles falsity. The system functioning is satisfactory but there appear aspects that might be improved in the future, nevertheless. Apart from technical implementation matters, this paper describes also security issues, and presents the user guide on how the software looks like from the user's point of view and how it might be used.

2. Introduction

Fake news are false stories that seem to be news that are usually used to mislead, manipulate and disinform. They are known since ancient history, but now they seem to overwhelm people and to be present everywhere. With the technology development and expanding availability of Internet, false information are easier to produce, distribute and spread, making it more complicated to correctly distinguish information that are worth to trust. Those news are published on specialized websites, social media platforms, or even as podcasts. According to a Pew Research Center survey (Walker and Matsa, 2021), about 50% adults living in the United States of America read and get news from social media. Every day, approximately 1.93 billion people are exposed to the information on the leading social media platform. According to the research done by Amy Watson, about 52% of United States citizens declared that they regularly spot fake news stories in 2018. It means that false information are ubiquitous and millions of people can be misled.

Many individuals have claimed that they have encountered fake news. According to Deloitte study (Auxier and Arbanas, 2021), 73% of United States news consumers feel alarmed when it comes to fake news. Nowadays, false information are easily spread via Internet, which means that they can reach plenty of people. By virtue of commonness of the problem and number of people it affects, our priority was to find an appropriate solution that will help people assess information easily. As a result, the browser extension that gets the content of a currently viewed article, sends it to the cloud function which determines whether passed data is fake or not, and returns the answer in the form of the browser alert, was created.

Our solution was intended to be simple to use, quick and reliable. It was accomplished almost as planned. The fake news detector in the form of the browser extension provides simplicity, as it requires only to click the extension's icon in order to launch the fake news detector. The process of assessing the article content takes approximately 10 seconds to receive the outcome, which is relatively long period of time. The reliability was achieved by 99% of the accuracy of the model used to evaluate articles.

The outcome of the project is recognized as a success, as all previously planned components were implemented. The fake news detector serves its purpose, providing the user with short, understandable response. Naturally, challenges during preparation of the project were inevitable. Struggles, that required the most attention, concerned several aspects of our implementation. Firstly, at the beginning, the group encountered issues regarding selection of the cloud provider. Financial aspects were the most significant in the first phase of the project. There was a lot of effort put also in the HTML parser, which required a look into the structure of every webpage to extract the content correctly. Issues related to the model concerned the choice of the library and of the classification algorithm. In this report, details concerning our solution will be thoroughly discussed.

3. State of art

Research was carried out in order to get to know what are the existing methods of fake news detection that would be inspirational while inventing the problem solution. In general, there exist two approaches, one of which concerns Machine Learning, whereas the second one concerns Neural Networks.

The goal of machine learning is to find patterns in data and use this to make estimates. It makes use of advanced algorithms in the learning process that parse data, learn from it, and eventually create a model. The model is then used to classify whether data provided by the user is fake or not. Several algorithms were analysed and compared in a research article "Fake News Detection Using Machine Learning Approaches" released by IOP Conf. Series: Materials Science and Engineering in 2021. These are: XGboost, Random Forests, Naive Bayes, K-Nearest Neighbours (KNN), Decision Tree, and SVM (Support Vector Machines. According to the report, the highest accuracy was obtained with use of XGboost algorithm, and it reached more than 75%. However, Neural Networks is a different approach, which was inspired by the human brain structure. It may be described as a web of interconnected entities wherein each of them is responsible for a simple computation.

As it turned out during the desk research, there are many possible implementation methods for Machine Learning and Neural Networks, however Python language provides developers with one of the most popular environments. There exist different Python libraries for Machine Learning and Neural Networks, including Scikit-learn, Pandas, and TensorFlow.

Another step was to search for existing solution for the fake news detection. Many different browser extensions were found. For instance, Fake News Detector, TrustedNews, and The Factual. A significant advantage of the last two mentioned solutions is that apart from simple news verification they also display objectivity and credibility expressed as a percentage. However, in case of The Factual it was noticed that the browser extension does not work properly, and it often crashes while opening.

Having the research done, the decision was made to base the project, that is the browser extension for fake news detection, on Neural Networks approach. The verification module was to be implemented with use of Python language and libraries Pandas and TensorFlow.

4. Project specification and requirements

4.1 Objectives

The main goal of the project was to provide a tool enabling the end user to easily and quickly inform the end user whether a journalistic article he wants to read on the website in the browser is true or false. The most appropriate solution was to create a browser plug-in as it meets all the requirements of the end user.

Design assumptions:

1. The browser plug-in will be written in JavaScript technology
2. Client-server architecture - the plug-in acts as a client, the whole classification of the article takes place on the server
3. Cloud services act as a server (supporting parallelization)
4. The classifier of the article is created on the basis of artificial intelligence (the classification is carried out on the basis of the obtained data)

Limitations:

1. The trial period offered by the cloud provider is used
2. Limited number of pages supported by the plugin - no resources (financial and human) to create an HTML parser capable of parsing websites of many publishers
3. Learning to use data containing US political articles published in English

4.2 Sources

1. Source of data used to train the model:
    a. Kaggle platform (link to dataset: https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset) - set of articles (title, content, type of the article and publish date). Most of the articles are from 2017 and focuses about American political news.
2. Google products
    a. As cloud provider Google company was chosen (reason why in point 3.3) - (documentation: https://cloud.google.com/docs)
    b. Other products
        - Google Colab Platform (documentation: https://colab.research.google.com/notebooks/basic_features_overview.ipynb)
        - TensorFlow (https://www.tensorflow.org/)

4.3 Tools and technologies

The following platforms and tools were used to implement the project:

1. Text classifier
    a. Dataset
        - Kaggle platform – High data availability, one of the most popular data sources.
    b. Data analysis (pre-processing)

        - Google Cloud Platform- a long trial period and wide availability of tools and the ease of creating virtual machines and using resources, the possibility of using TPU processors, the possibility of using resources by the google colab platform.
        - Google Colab Platform - enabling the use of cloud resources, access to python libraries, ease of use (possibility of calling the code step by step)
        - Python language (3.7 version) - wide availability of pre-processing and machine learning libraries
        - Pandas library - reading data from a CSV file and their easy and quick processing, possibility of connecting with the TensorFlow library
        - Scikit-learn library – splitting sets into train and validation set
    c. Machine learning model
        - TensorFlow library - creating neural networks and training them, a wide possibility of analysing learning results.
        - Tensorflow_hub library - use of already well-trained neural networks by google (possibility of adding to your architecture)
        - Tensorboard library - visualization of learning outcomes
2. Browser plug-in (client-server architecture realization)
    a. Plug-in (client)
        - JavaScript - the ability to write browser plugins (Chromium-based browser) and send and receive data from the server
    b. Server
        - Cloud function in the google cloud - serverless code calling and parallelization (many connected clients).

4.4 Skills

The implementation of the project in a group, above all, required the ability to organize and divide work, as well as knowing the strengths and weaknesses of each member. The most important and time-consuming task was to train the text classifier in such a way that the obtained result would be satisfactory for us and would be realistic in production conditions. For this, skills in the field of data analysis, knowledge of the theory of data pre-processing, as well as the specifications of various machine learning algorithms (to initially reject those that are definitely not suitable) were required. The project also needed to know Python and JavaScript, know the basics of networking and have knowledge of tools and cloud solutions.

4.5 Risk

The main risk of failure was ignorance and indecision as to how and with what technologies a text classifier should be trained, but with appropriate mobilization and obtaining the necessary knowledge, this risk was avoided.

Our project still faces the risk of failure in production conditions due to:

1. Cloud trial period - Limited resources and limited availability period

2. Lack of knowledge how the model will behave in production conditions - lack of an appropriate test set.

3. It is not known how the data extraction will behave with different HTML codes (implementation details in point: Solution Description, point 2.)

4.6 Measures of success

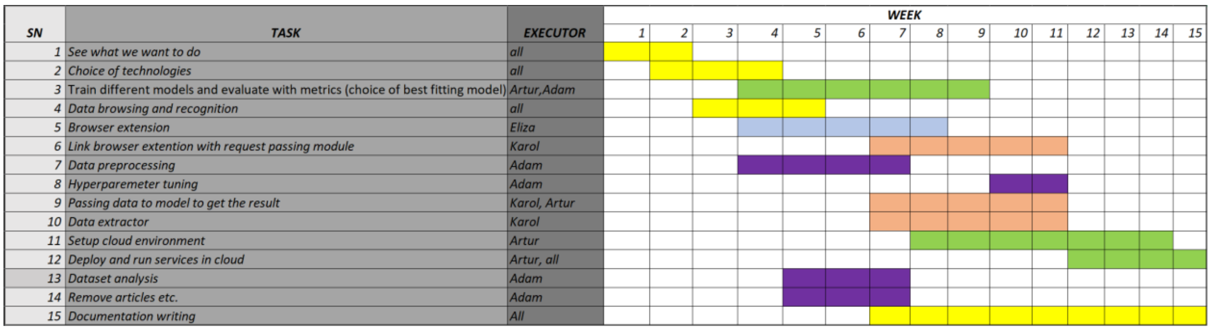
The success of a project must be measured with several metrics.

1. Working prototype - The implemented prototype works correctly, in accordance with the initial assumptions described in section 3.1, i.e. Client sends data downloaded from the website after starting the plug-in to the server and receives a response from the server.

2. Evaluation of the text classifier - Evaluation of the model based on the accuracy and F1-score on the validation set. In real conditions, a test set simulating the requests of clients reading various articles should be created, but it was not possible due to the lack of human resources.

5. Management

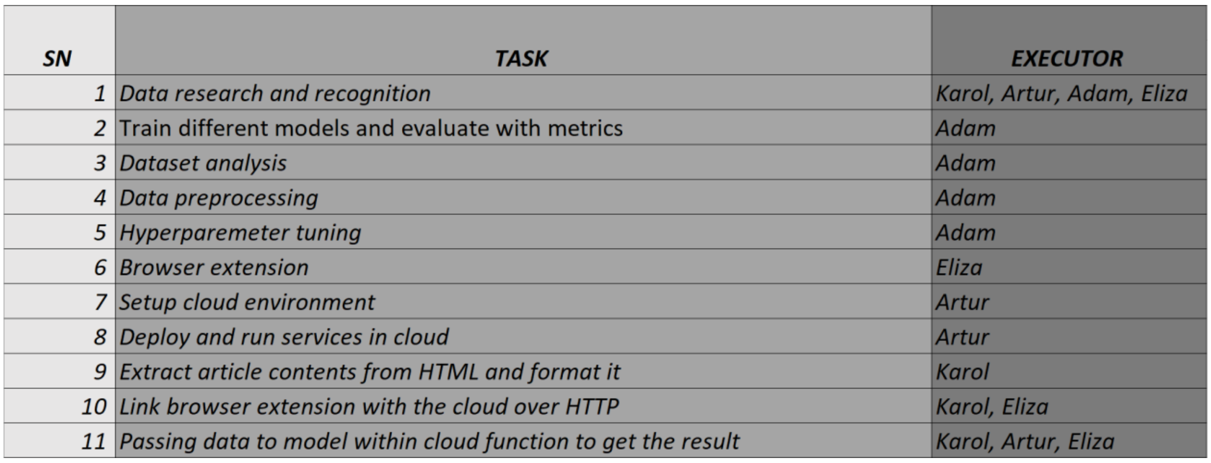
The architecture of the project is simple which allowed to easily divide the work between members of the project group and focus on the principle of working. At early stages of the project, when we were planning and deciding what will be done a schedule was created to approximate what everyone will be taking care of and in what sequence it will be performed. It can be seen on Fig. 1.



SN	TASK	EXECUTOR	WEEK														
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	*See what we want to do*	*all*	🟨	🟨													
2	*Choice of technologies*	*all*		🟨	🟨	🟨											
3	Train different models and evaluate with metrics (choice of best fitting model)	*Artur,Adam*				🟩	🟩	🟩	🟩	🟩	🟩						
4	*Data browsing and recognition*	*all*			🟨	🟨	🟨										
5	*Browser extension*	*Eliza*				🟦	🟦	🟦	🟦	🟦							
6	*Link browser extention with request passing module*	*Karol*							🟧	🟧	🟧	🟧	🟧				
7	*Data preprocessing*	*Adam*				🟪	🟪	🟪	🟪								
8	*Hyperparemeter tuning*	*Adam*										🟪	🟪				
9	*Passing data to model to get the result*	*Karol, Artur*							🟧	🟧	🟧	🟧	🟧				
10	*Data extractor*	*Karol*							🟧	🟧	🟧	🟧	🟧				
11	*Setup cloud environment*	*Artur*								🟩	🟩	🟩	🟩	🟩	🟩	🟩	
12	*Deploy and run services in cloud*	*Artur, all*												🟩	🟩	🟩	🟩
13	*Dataset analysis*	*Adam*					🟪	🟪	🟪								
14	*Remove articles etc.*	*Adam*					🟪	🟪	🟪								
15	*Documentation writing*	*All*							🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨

Fig. 1. Initial schedule with division of work between members time and task-wise.

Although in reality, we haven't hold up to the schedule time – wise, it still describes pretty well, what individual team members have done in order to create working project. The complete division of work between members can be seen below on Fig. 2.



SN	TASK	EXECUTOR
1	*Data research and recognition*	*Karol, Artur, Adam, Eliza*
2	Train different models and evaluate with metrics	*Adam*
3	*Dataset analysis*	*Adam*
4	*Data preprocessing*	*Adam*
5	*Hyperparemeter tuning*	*Adam*
6	*Browser extension*	*Eliza*
7	*Setup cloud environment*	*Artur*
8	*Deploy and run services in cloud*	*Artur*
9	*Extract article contents from HTML and format it*	*Karol*
10	*Link browser extension with the cloud over HTTP*	*Karol, Eliza*
11	*Passing data to model within cloud function to get the result*	*Karol, Artur, Eliza*

Fig. 2. Work division between members.

In short:

- Adam handled data analysis and model creation
- Karol handled article contents extracting from HTML and linking the extension with the model over HTTP and cloud functions
- Artur handled setup and configuration of the cloud, and creating the cloud function itself
- Eliza handled creation of browser extension and reading webpage HTML for it to be analysed and formatted

6. User guide

The browser extension itself is very simple to use. In order to setup it in the browser one needs to unpack folder with three necessary extension files. To setup it in Google Chrome following steps need to be performed:

1. Enter Extensions tabs in Chrome.

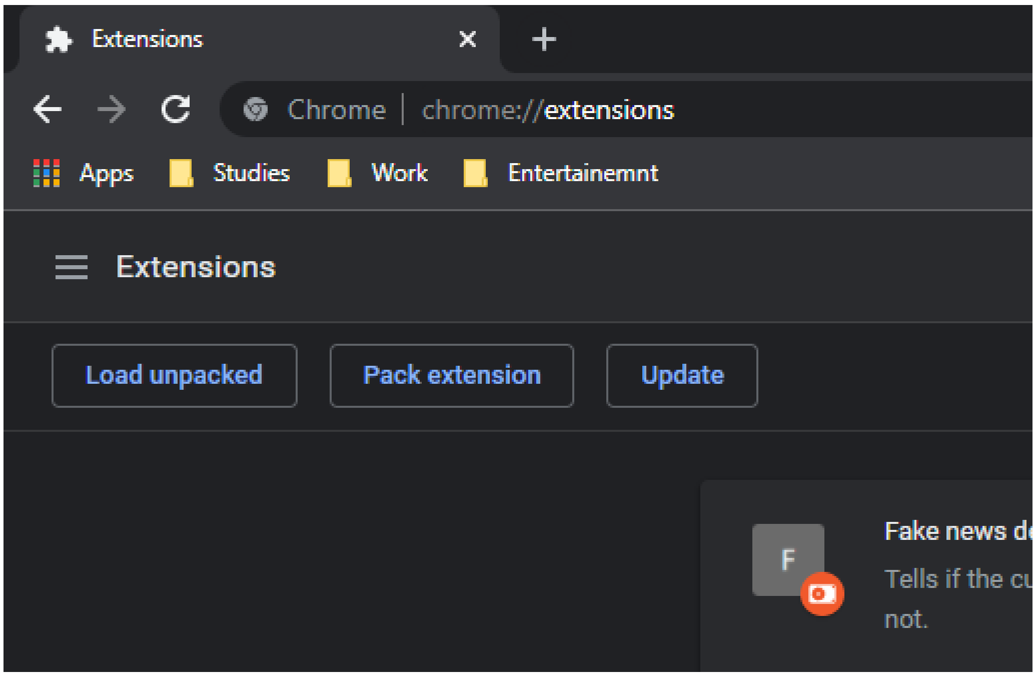


Fig. 3. Extension panel in Google Chrome.

2. Press on “Load unpacked” button at the left top of the browser’s window and select the “extension” folder from project’s root folder. It has to contain “background.js”, “content.js” and “manifest.json”.

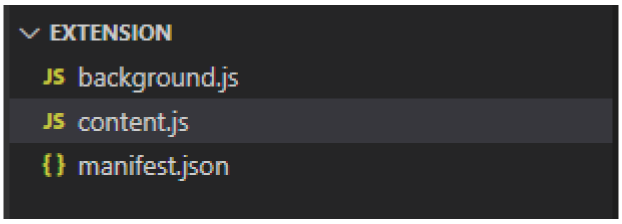
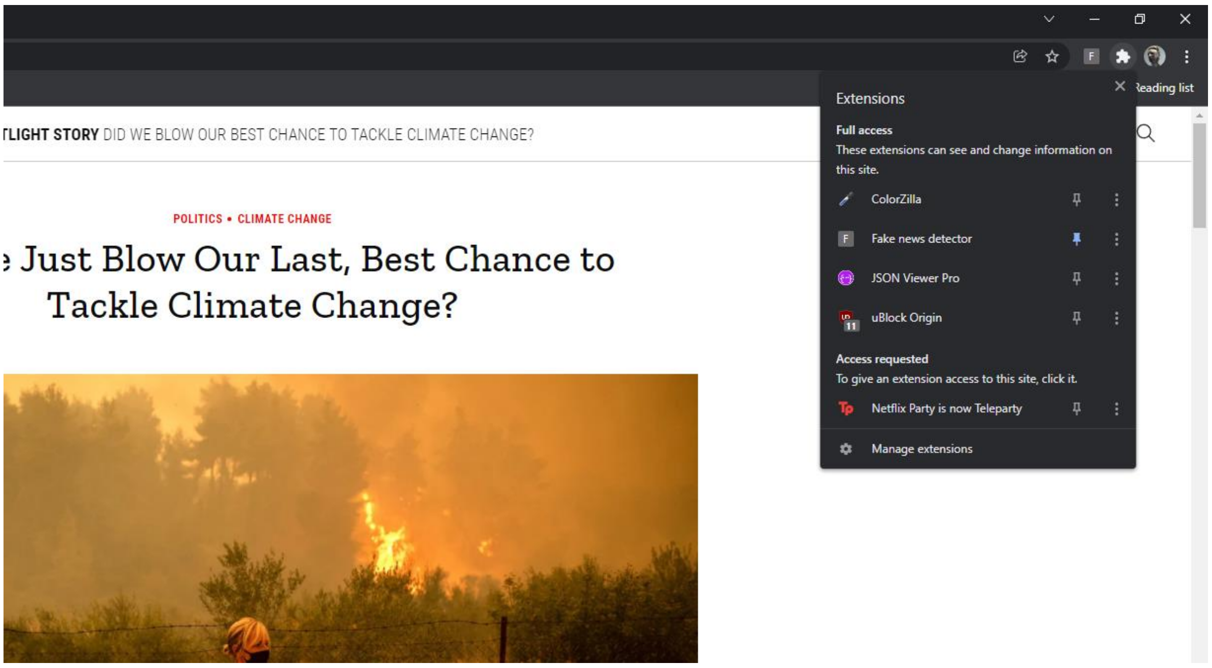


Fig. 4. Contents of extension folder.

3. The extension should now be loaded into the browser and fully operational. In order to use it, one has to enter a site with article which he want to determine as fake or not, and press the extension from the installed extensions tab.


TLIGHT STORY DID WE BLOW OUR BEST CHANCE TO TACKLE CLIMATE CHANGE?

POLITICS • CLIMATE CHANGE

e Just Blow Our Last, Best Chance to Tackle Climate Change?

Extensions

Full access

These extensions can see and change information on this site.

ColorZilla

Fake news detector

JSON Viewer Pro

uBlock Origin

Access requested

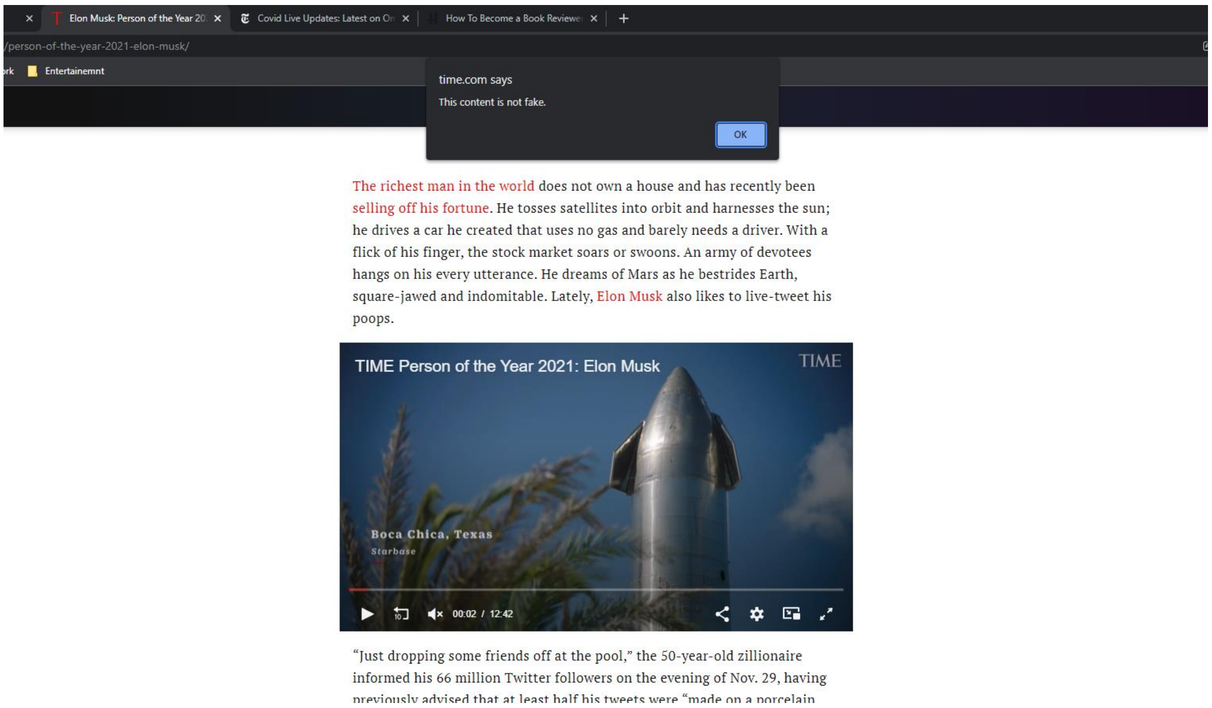
To give an extension access to this site, click it.

Netflix Party is now Teleparty

Manage extensions

Fig. 5. Fake news detector should be visible in the list accessible in the right top corner of the browser

4. Assuming everything went fine an alert should be shown in the browser with message “This content is not fake” or “This content is fake”. There is also a chance that an alert with message “Incorrect webpage format. Are you sure this is an article?” This may happen when the extension cannot process the HTML of the webpage correctly.


Elon Musk: Person of the Year 20 | Covid Live Updates: Latest on Or | How To Become a Book Reviewe

/person-of-the-year-2021-elon-musk/

Entertainemnt

time.com says

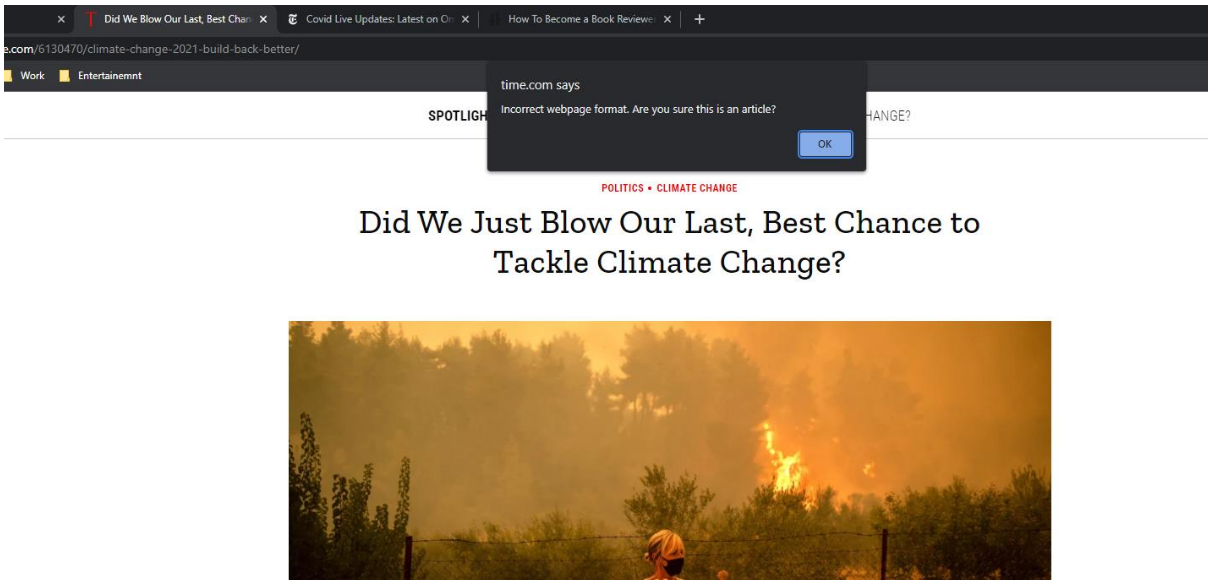
This content is not fake.

OK

The richest man in the world does not own a house and has recently been selling off his fortune. He tosses satellites into orbit and harnesses the sun; he drives a car he created that uses no gas and barely needs a driver. With a flick of his finger, the stock market soars or swoons. An army of devotees hangs on his every utterance. He dreams of Mars as he bestrides Earth, square-jawed and indomitable. Lately, Elon Musk also likes to live-tweet his poops.

“Just dropping some friends off at the pool,” the 50-year-old zillionaire informed his 66 million Twitter followers on the evening of Nov. 29, having previously advised that at least half his tweets were “made on a porcelain

Fig. 6. Example response when the article was determined as not fake


Did We Blow Our Last, Best Chan
Covid Live Updates: Latest on Om
How To Become a Book Reviewer
e.com/6130470/climate-change-2021-build-back-better/
Work
Entertainemnt
SPOTLIGH
HANGE?
POLITICS • CLIMATE CHANGE
Did We Just Blow Our Last, Best Chance to Tackle Climate Change?

time.com says

Incorrect webpage format. Are you sure this is an article?

OK

Fig. 7. Example response when the webpage article cannot be processed

7. Solution description

7.1 Plug-in (client)

Plug-in consists of three files: manifest.json, background.js, and content.js.

Manifest.json contains important information such as permissions that extension needs, description or background files it consists of, actions it performs. In our case, permissions to access the active tab and scripting were granted.

Background.js contains function which listens for click on the extension's icon and runs the content.js file.

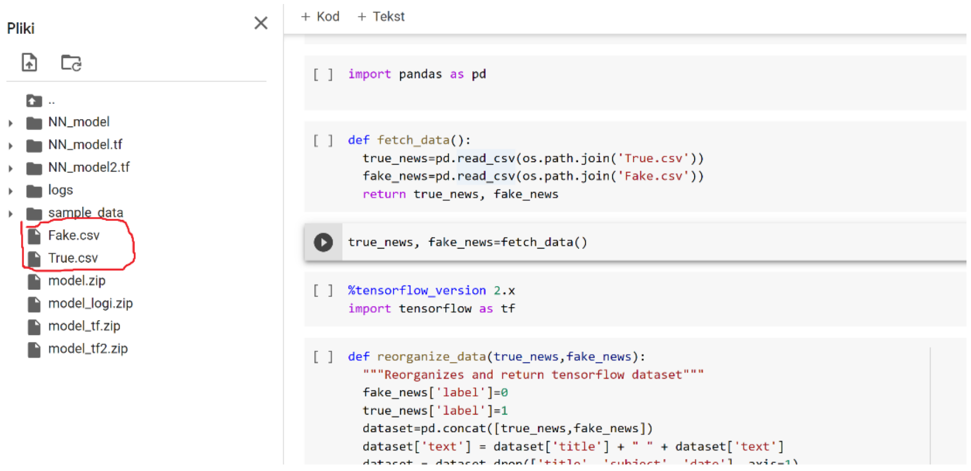
Content.js is the longest file that consists of HTML parser, function sending POST request to the server, and function retrieving content from currently opened tab.

a. GET function - this function is responsible for retrieving the webpage in the form of the HTML code by means of HTTP GET request.
b. HTML parser – the HTML retrieved from the currently opened webpage needs to be parsed so that only the relevant article text will be sent to the model. In order to do that an attempt is made to find an <article> tag. Then its textual contents are read and the text is parsed so that all the tags like <div>, <p> or <a> as well as theirs attributes are removed. Then the text is formatted, so that all whitespaces are deleted, and the text actually looks like an article that was written by a real person and is ready to be sent to the model.
c. POST function – this function sends the data to the server as HTTP POST request and returns a response from the server, which indicates whether the sent data was assessed as fake or not.

7.2 Model

7.2.1 Pre-processing of data

As mentioned in point 3.2 data obtained from Kaggle platform were stored in 2 files: True.csv and Fake.csv



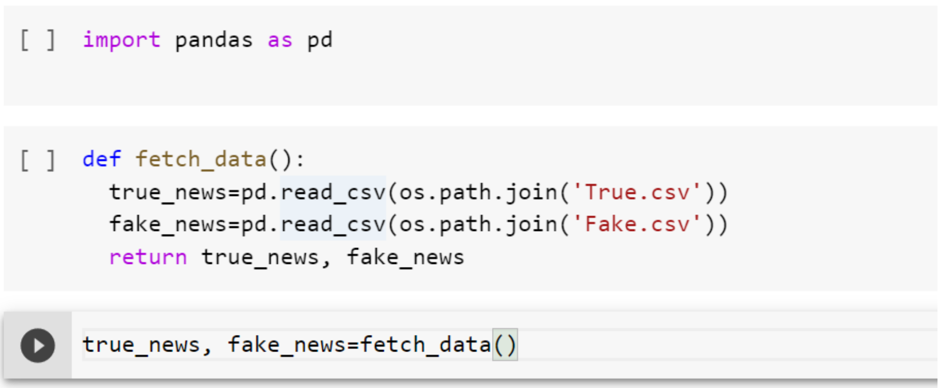
```
import pandas as pd
```

```
def fetch_data():
    true_news=pd.read_csv(os.path.join('True.csv'))
    fake_news=pd.read_csv(os.path.join('Fake.csv'))
    return true_news, fake_news
```

```
true_news, fake_news=fetch_data()
```

```
%tensorflow_version 2.x
import tensorflow as tf
```

```
def reorganize_data(true_news,fake_news):
    """Reorganizes and return tensorflow dataset"""
    fake_news['label']=0
    true_news['label']=1
    dataset=pd.concat([true_news,fake_news])
    dataset['text'] = dataset['title'] + " " + dataset['text']
    dataset = dataset.drop(['title', 'subject', 'date'], axis=1)
```



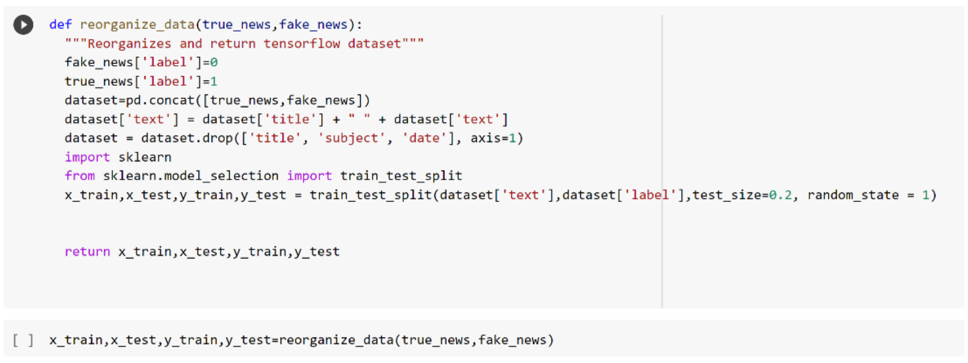
```
import pandas as pd
```

```
def fetch_data():
    true_news=pd.read_csv(os.path.join('True.csv'))
    fake_news=pd.read_csv(os.path.join('Fake.csv'))
    return true_news, fake_news
```

```
true_news, fake_news=fetch_data()
```

Having defined fetch_data function with use of Pandas library it was possible to store data with dataframe objects.

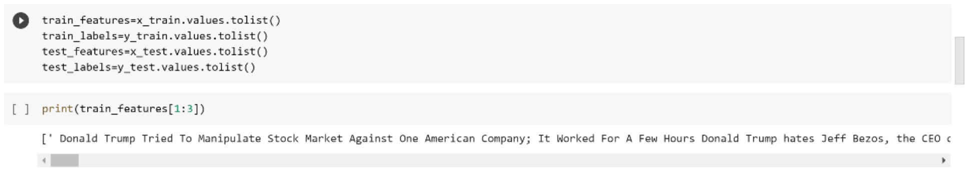
As needed for data pre-processing it was necessary to label positive samples and negative samples, merge title with the article content and drop irrelevant data (date and subject). Positive and negative samples were merged into one dataframe object, and then they were split into training and validation (test) sets (labels and features are stored in different objects). Function provided by scikit-learn library (train_test_split) allowed to shuffle the data and split with given size of validation set (in this case 20%). All was done with use of function reorganize_data.



```
def reorganize_data(true_news,fake_news):
  """Reorganizes and return tensorflow dataset"""
  fake_news['label']=0
  true_news['label']=1
  dataset=pd.concat([true_news,fake_news])
  dataset['text'] = dataset['title'] + " " + dataset['text']
  dataset = dataset.drop(['title', 'subject', 'date'], axis=1)
  import sklearn
  from sklearn.model_selection import train_test_split
  x_train,x_test,y_train,y_test = train_test_split(dataset['text'],dataset['label'],test_size=0.2, random_state = 1)


  return x_train,x_test,y_train,y_test
```

```
[ ] x_train,x_test,y_train,y_test=reorganize_data(true_news,fake_news)
```

In order to be able to input this data to the TensorFlow neural network it was necessary to convert data to python list objects.

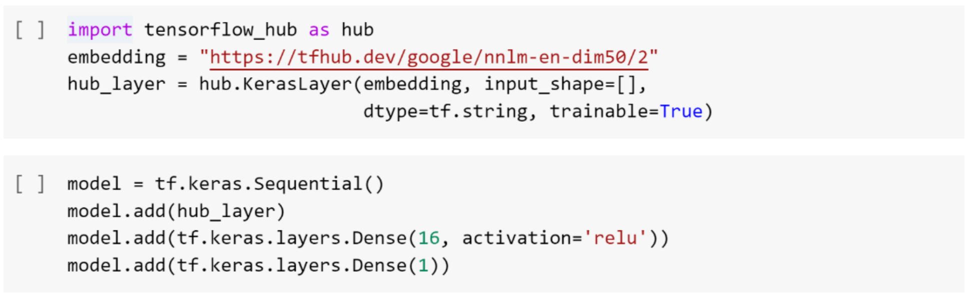


```
train_features=x_train.values.tolist()
train_labels=y_train.values.tolist()
test_features=x_test.values.tolist()
test_labels=y_test.values.tolist()
```

```
[ ] print(train_features[1:3])

['  Donald Trump Tried To Manipulate Stock Market Against One American Company; It Worked For A Few Hours Donald Trump hates Jeff Bezos, the CEO c
```

7.2.2 Model training

As for text classifier Neural Network was chosen as the best way to face the NLP problem. The main reason of usage is huge flexibility and possibility to link with external architectures (models), which was used this case.



```
[ ]  import tensorflow_hub as hub
     embedding = "https://tfhub.dev/google/nnlm-en-dim50/2"
     hub_layer = hub.KerasLayer(embedding, input_shape=[],
                                dtype=tf.string, trainable=True)
```

```
[ ]  model = tf.keras.Sequential()
     model.add(hub_layer)
     model.add(tf.keras.layers.Dense(16, activation='relu'))
     model.add(tf.keras.layers.Dense(1))
```

As visible in the screenshot above, the embedding layer is so called hub layer. Our model uses *nnlm-en-dim50* (pretrained neural network managing the embedding and tokenizing data. More in point 3.2). Our model contains one more hidden layer, and output layer with softmax activation function.

The summary of the model is as following:

```
model.summary()
```

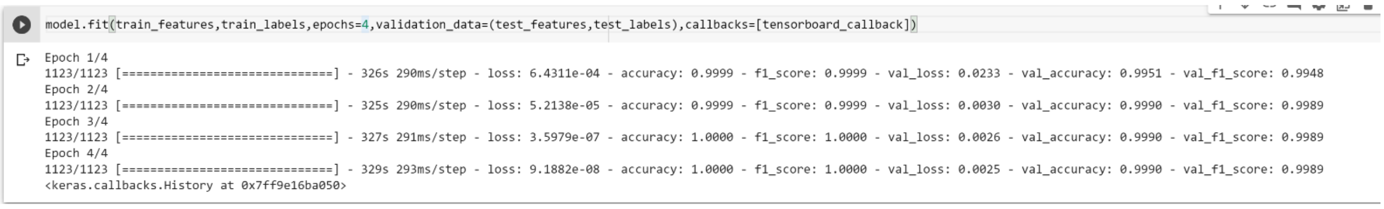
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer (KerasLayer)     (None, 50)                48190600
_________________________________________________________________
dense (Dense)                (None, 16)                816
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 17
=================================================================
Total params: 48,191,433
Trainable params: 48,191,433
Non-trainable params: 0
_________________________________________________________________
```

As for future evaluation F1-score function was defined, because it is not provided by TensorFlow.

```python
import keras.backend as K

def f1_score(y_true, y_pred):


    c1 = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    c2 = K.sum(K.round(K.clip(y_pred, 0, 1)))
    c3 = K.sum(K.round(K.clip(y_true, 0, 1)))


    if c3 == 0:
        return 0.0


    precision = c1 / c2


    recall = c1 / c3


    f1_score = 2 * (precision * recall) / (precision + recall)
    return f1_score
```

```python
model.compile(optimizer='adam',
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy', f1_score])
```

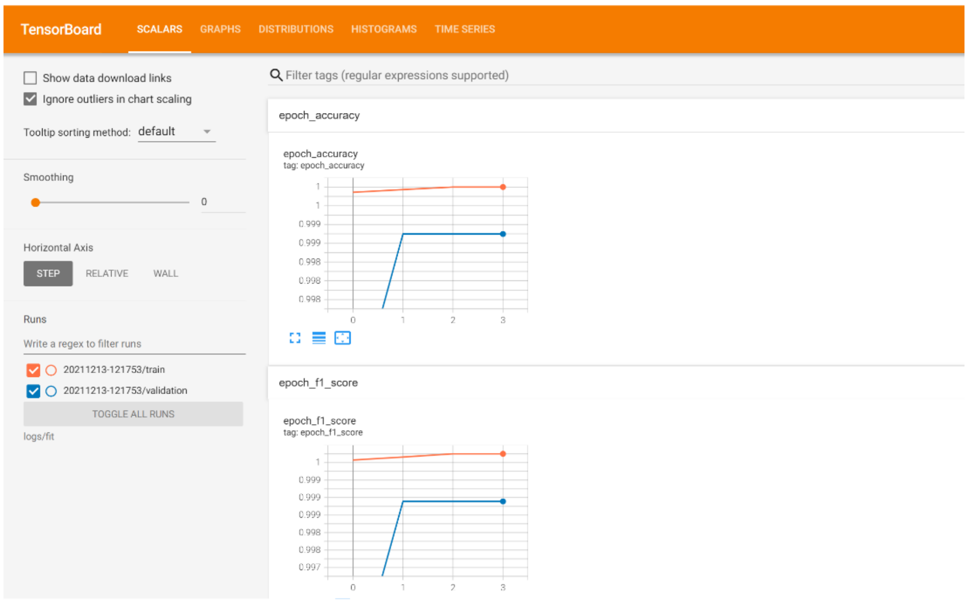
Model was trained with 4 epochs (longer weight update was not necessary).



```
model.fit(train_features,train_labels,epochs=4,validation_data=(test_features,test_labels),callbacks=[tensorboard_callback])
```

```
Epoch 1/4
1123/1123 [==============================] - 326s 290ms/step - loss: 6.4311e-04 - accuracy: 0.9999 - f1_score: 0.9999 - val_loss: 0.0233 - val_accuracy: 0.9951 - val_f1_score: 0.9948
Epoch 2/4
1123/1123 [==============================] - 325s 290ms/step - loss: 5.2138e-05 - accuracy: 0.9999 - f1_score: 0.9999 - val_loss: 0.0030 - val_accuracy: 0.9990 - val_f1_score: 0.9989
Epoch 3/4
1123/1123 [==============================] - 327s 291ms/step - loss: 3.5979e-07 - accuracy: 1.0000 - f1_score: 1.0000 - val_loss: 0.0026 - val_accuracy: 0.9990 - val_f1_score: 0.9989
Epoch 4/4
1123/1123 [==============================] - 329s 293ms/step - loss: 9.1882e-08 - accuracy: 1.0000 - f1_score: 1.0000 - val_loss: 0.0025 - val_accuracy: 0.9990 - val_f1_score: 0.9989
<keras.callbacks.History at 0x7ff9e16ba050>
```

As for visualization of learning process Tensorboard was used. As visible after one epoch model achieved almost its maximum accuracy and F1_score.



☐ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing 0
☑ 20211213-121753/train
☑ 20211213-121753/validation

Model after 3 epochs reached 100% accuracy and 99.9% F1-score on training set and 99.9% accuracy and 99.9% F1-score on validation set. It means that clear separation can be observed between positive and negative samples and the model copes very well with text classification (the architecture is adapted to the needs).

7.2.3 Model storage

Model was saved with .tf extension for later usage.

7.3 Cloud deployment (server)

The fake news detection system was deployed with use of Google Cloud Function platform. Its advantage over other cloud systems (for instance, Google Compute Engine offering virtual machines hosting) is that functions set up there get triggered when an event attached to it is fired and terminates after execution of the function. Thus, it is an effective solution.

Before cloud function was created and configured, variables.index and variables.data-00000-of-00001 files were uploaded to Google Cloud Storage Bucket. The first file stores the list of variable names, whereas the second one stores the actual values of all the variables saved. Then the cloud function was created in such a way that HTTP is its trigger type, and it allows unauthenticated invocations. When it comes to memory allocated by the function, it was set to be 4 gigabytes due to size of variables.data-00000-of-00001 the cloud function operates on. The source code of the cloud function consists in two files. The first one is main.py python file, where the function code is stored and is executed when the trigger event happens. The second file, requirements.txt, is a place, where libraries required for function execution are declared.

In response to an HTTP request for model prediction, the is_fake entry point of the cloud function is called and article content from browser extension is retrieved. Then the process of model loading starts. To do so the function loads variables form cloud storage and stores them in a form of Blob objects. Next, variables are passed to the model, which is now ready to make a prediction on article falsity. If data sent by the browser extension is not empty, the model proceeds with the prediction and returns a softmax result. Otherwise, the function returns information that nothing was sent for prediction. The softmax results enables the final stage of the cloud function action. If its value is lower or equal to 0, the function returns "False" information. When its greater than 0, then the returned information is "True".

8. Security

Google Cloud Platform provides variety of security aspect for the Cloud Functions. In our solution, access control based on authentication was managed. In order to provide the necessary functionality to our browser extension, public access was set. It means that all users have a possibility to invoke the fake detection function. They do not have access to other aspects of our cloud solution. Also, there is a possibility to fully control all the permissions, i.e. editors, owners and service agents and to control to which resources the cloud function has access.

9. Results discussion

The results of the model on articles in practice are very satisfactory. The datasets had a lot of controversial articles revolving around US politics. Because of that, in the completed project tests, there is a tendency where articles about politics are more often identified as fake. But it as well may be caused by the fact that this kind of topics raise often more controversy and division that other topics e.g. events' summaries, celebrity descriptions etc. Generally though, the results yielded by the model are good.

The main advantage to our solution is its ease of use, as the only thing required to verify the truthfulness of an article is a press of a button. There is no need to enter any other website, paste website URLs etc.

There are two main weak points of this solution, first of which is the time it takes to get the result. Although it is very quick to activate the extension, we need to wait several seconds for the results. The wait time hardly ever reaches more than 10 seconds, but it is still not user friendly. The reason for that is the fact that in the cloud function the model gets downloaded each time the function is called. Apart from that, the extension and HTML article extraction is quick.

The other weak point of the app is the fact that sometimes the HTML cannot be processed even though the site is an article. Various sites are built differently when it comes to HTML structure. In HTML5 an <article> tag has been introduced, which in theory should be used to contain articles or longer text contents. Although it is used on most sites, there are some, where the structure is different and with our implementation of parsing the contents of <article> tag, the article cannot be extracted and formatted. An alternative approach would be to focus only on the most established sites like 'The Times', 'New York Times' or CNN and extract the articles depending on the site from which data is extracted, but for this project a more general approach was chosen.

10. Summary

Summarizing, the team managed to complete the project on fake news detection, which resulted in creation of the browser extension that evaluates online articles for being fake or true. Main design points of the project include:

- Browser extension – extracts the article content from HTML file, sends it as a request to cloud computing system, waits for the result and after receiving it, displays the information on article falsity to the user
- Classifying model based on Neural Networks – already trained model that makes predictions on article falsity basing on the parameters put to the model by cloud function. The parameters were obtained in the process of predefined data set analysis
- Google cloud function – a serverless environment that reacts to request send by the browser extension. In a request it receives an article content that is passed by the function to the loaded model, and then the obtained result is returned from the cloud to the user via browser extension

As for the project perspectives, there are aspects that might be improved in the future. User interface, article recognition and execution time may be mentioned. The first aspect concerns the way how the browser extension looks like for the user. The interface might be developed to be more eye-catching, because in the first project stage it was not the priority in the teamwork. Another aspect is related to the fact that the software does not work for websites that does not include an <article> tag in their HTML structure. Therefore, extraction of content despite lack of such tag might be added. And the last-mentioned aspect concerns one of the biggest weaknesses of the current system, that is long time needed in order to get the answer for the request from the cloud function. To improve that, the model loading might need to be reorganised.

Eventually, the project affected knowledge and skills of the team members. Especially abilities related to management of cloud computing systems (including setting up cloud environment and reacting to exterior requests), machine learning and neural networks (model training, accuracy control, and final usage), writing extensions for browsers, and extracting specific data from HTML files (which required knowledge acquirement from JavaScript and JSON branch). Additionally, the work on project resulted in improvement of soft skills, including communication, work in group, creativity, and work management. However, there are plenty of other skills that were developed and are difficult to notice, but they will positively affect future work of each team member.

11. References

"fake news", no date, in *Cambridge Dictionary*, Dictionary.Cambridge.org. (accessed 1 January 2022)

Arbanas, Auxier, ‘Majority of news consumers see “fake news” as a big problem today’, *Deloitte*, 2021, p. 1, https://www2.deloitte.com/us/en/insights/industry/technology/study-shows-news-consumers-consider-fake-news-a-big-problem.html (accessed 3 January 2022)

Kalsnes, B., ‘Fake News’, *Oxford Research Encyclopedias*, 2018, p. 1, https://oxfordre.com/communication/view/10.1093/acrefore/9780190228613.001.0001/acrefore-9780190228613-e-809 (accessed 1 January 2022)

Matsa, Walker, ‘News Consumption Across Social Media in 2021’, *Pew Research Center*, 2021, p. 1, https://www.pewresearch.org/journalism/2021/09/20/news-consumption-across-social-media-in-2021/ (accessed 2 January 2022)

Statista Research Department ‘Facebook: number of daily active users worldwide 2011-2021’, *Statista*, 2021, p. 1, https://www.statista.com/statistics/346167/facebook-global-dau/ (accessed 1 January 2022)

Watson, A., ‘Frequency of online news sources reporting fake news U.S. 2018’, *Statista*, 2019, p. 1, https://www.statista.com/statistics/649234/fake-news-exposure-usa/ (accessed 1 January 2022)

Khanam, Alwasel, Sirafi, Rashid, ‘Fake News Detection Using Machine Learning Approaches’, IOPScience, 2021