Browser Extension for Detection of Fake News and Disinformation

Lumbardha Hasimi and Aneta Poniszewska-Marańda(✉)

Institute of Information Technology, Lodz University of Technology, Lodz, Poland
lumbardha.hasimi@dokt.p.lodz.pl, aneta.poniszewska-maranda@p.lodz.pl

Abstract. Fake news is information usually used to mislead, manipulate or disinform while reaching a certain audience and going viral in rather a short period. Currently it started to be the bigger and bigger problem in Internet, mass media and in everyday life. The pervasive and wide-spreading effect of fake news content is becoming a serious concern of our era. Considering the emergent need for research in this area, our work aims to observe, analyse and propose a solution to the fake news topic. This work presents an Internet browser extension, aiming to notify the user regarding the credibility of the information and carrying out fake news detection. The system as an extension is designed using JavaScript environment, cloud function configured on Google Cloud platform while using Neural Network Model based on TensorFlow library for predictions process on the credibility of the content. The paper also discusses and presents approaches and models on fake news detection, and subsequently security issues on the system's functionality.

Keywords: fake news detection · cloud computing · browser plugin · prediction of text · neural networks

1 Introduction

Fake news is information usually used to mislead, manipulate or disinform while reaching a certain audience and going viral in rather a short period. It is almost certain that the spread of fake news has had a grave impact on social cohesion, democratic processes and most of all has raised serious concerns among different entities.

In the midst of the great challenge of identifying, analysing, and understanding this phenomenon and many underlying processes, there have been proposed and developed many different approaches and techniques. Although these approaches differ in the choice of algorithmic techniques and adaptation, still there is a common share of techniques in the methodology and deployment [1].

With the recent developments in technology and the internet advances, false information is easier to create, share and spread, making it more complicated to correctly distinguish it from true information. Such news are published on

M. Papadaki et al. (Eds.): EMCIS 2022, LNBIP 464, pp. 209–220, 2023.
https://doi.org/10.1007/978-3-031-30694-5_16

specialized websites, social media platforms or even as podcasts. The multimodal nature of fake news is making the detection process even more challenging, though it has shown more evidence of the happening of news events and presenting new opportunities to detect features in fake news [2].

Every day, approximately 1.93 billion of people are exposed to information on the leading social media platform. The most recent example of this is the COVID-19 pandemic – almost 80% of consumers in the United States reported having seen fake news on the coronavirus outbreak, highlighting the extent of this issue and the reach fake news can achieve [3].

False information is ubiquitous, and millions of people can be misled in a matter of seconds. Under the commonness of the problem and the number of people it affects, our work aimed to find an appropriate solution that will help people assess information easily. As a result, we proposed a detection system as a browser extension that gets the content of a currently viewed article, sends it to the cloud function, to determine the veracity of the content, and returns the answer in the form of the browser alert.

This solution was intended to be quick, simple, and reliable. The fake news detector in the form of the browser extension provides simplicity, as it requires only a click in the extension to launch the fake news detector. The process of assessing the article content and receiving an outcome takes approximately ten seconds, a feature that needs further enhancement. The reliability of the accuracy of the model used to evaluate the content of the articles reached up to 99%. The fake news detector serves its purpose, providing the user with a short and comprehensible response.

This paper presents the proposal of browser extension for fake news and disinformation detection, analysing the aspects regarding the implementation, design, deployment in the cloud, libraries, algorithms, classification and data processing. The paper is structured as follows: Sect. 2 presents the related work in the field, the existing solutions and the innovations available. Section 3 describes the methodology, basic architecture and model's design. Section 4 deals with the cloud deployment while Sect. 5 showcases the plugin, evaluation and results.

2 Related Works

Research was carried out to learn about the current state of the art, existing methods, and research productivity. In general, the literature and existing solutions revolve around the use of Machine Learning and Neural Networks use in fake news detection.

Machine learning as a method concerns ways of finding patterns in data and using it to make estimations [4]. With the help of advanced algorithms in the learning process, it is possible to create models used to classify data provided by the user as fake or not. In the study [5] about the use of machine learning approaches in fake news detection, several algorithms and techniques were analysed and compared. According to the report, among algorithms such as XGboost, Random Forests, Naive Bayes, K-Nearest Neighbours (KNN), Decision

Tree, and SVM, the highest accuracy was obtained with XGboost algorithm, which was higher than 75%.

Neural networks, on the other hand, inspired by the human brain structure might be described as a web of interconnected entities, each of them responsible for a simple computation [6]. Furthermore, there are many possible implementation methods for machine learning and neural networks, for which Python language provides developers with one of the most popular environments. Some of the most commonly known libraries in Python include Scikit-learn, Pandas and TensorFlow.

Existing solutions as a browser extension for fake news detection include *Check-it* [1], *FakerFact* [7], *BRENDA* [8], *TrustedNews* [9], *The Factual* [10]. A significant advantage of the last two solutions is that apart from simple news verification they also display objectivity and credibility expressed as a percentage. Nevertheless, in the case of *The Factual* [10], it was noted that the browser extension often crashes while opening, perceiving also not proper functionality. Furthermore, browser extensions such as *The Factual*, *TrustedTimes* and *FakerFact*, although claimed to support automated fact-checking and being listed in the Google extension store, there is no available documentation of the models used. Moreover, it was not found to have any system which can narrow down the claim within the article using fact-checkworthiness detection and use that claim to detect fake news.

However, *BRENDA* [8] solution as a browser extension for fake news detection provides many feature evidence at the both-word level and the sentence level. It follows a client-server architecture and has a frontend and backend module, where the frontend is a browser extension and the backend is a python Flask server. This way as a solution, it stands out compared to the above-mentioned solutions.

Check-it [1], on the other hand, effectively combines a set of diverse signals as a form of the pipeline, to accurately classify fake news articles and inform the user, while ensuring user's privacy and easy experience. The system contains four main components, including matching, similarity checking and comparing, analysing user behaviour in social networks, and classifying linguistic features using different feature engineering processes [1]. Furthermore, most of the works above employ server-side APIs with constant communication, utilize HTTP cookies, request permissions, and require account registration. These actions are taken to have better results and higher accuracy in identifying fake news but may also have a negative impact making users reluctant in using it on browsing routine. Considering everything above, it was decided on browser extension, using the Neural Networks approach. The verification module is implemented with the use of Python language and libraries Pandas and TensorFlow.

3 Proposed Fake News and Disinformation Detection Solution

Aiming to tackle the issue of fake news and disinformation detection, this paper presents the proposal of the system on article veracity, that consists of three

main components. The browser extension extracts the article content from the HTML file, sends it as a request to the cloud computing system, after receiving a result, it displays to the user the information on article authenticity.

Secondly, the classifying model based on Neural Networks – a trained model that makes predictions on article authenticity based on the parameters put to the model by cloud function. The parameters were obtained in the process of pre-defined data set analysis, whereas Google cloud as a serverless environment reacts to the request sent by the browser extension. In a request, it receives an article content that is passed by the function to the loaded model, and finally, the obtained result is returned from the cloud to the user through the extension.

The proposed solution aims to significantly improve the quality of articles and information that are served to the reader, and this way hinders the fake news spread and dissemination of information. Building the insights of the system, different tools were engaged to reach the objective of veracity and credibility of the content. For the data analysis stage, the pre-processing phase, using Google Cloud Platform through all the available tools and libraries, engaged libraries such as Pandas and Scikit-learn. For the machine learning model, optimized we choose TensorFlow to create and train neural networks, hence carried out by *Tensorboard* library for visualization of learning outcomes. Finally, JavaScript for the plug-in and google cloud employs serverless code calling and parallelization.

To make it possible for the end-user to have a quick warning regarding the content credibility of the article, a browser plug-in was seen as an apt solution. Thus, it is proposed the plug-in acting as a client, whereas the entire classification procedure takes place on the server, namely cloud service supporting parallelization. The classification itself contains artificial intelligence tools carrying out the process based on the obtained data. There are three main files, consisting of *manifest.json*, *background.js* and *content.js* Manifest.

Manifest.json contains important information such as permissions that extension needs, description, or background files which consist of actions it performs. In this case, permissions to access the active tab and scripting were granted. Background.js contains a function that listens for click on the extension's icon and runs the *content.js* file. *Content.js* is the longest file that consists of an HTML parser, function sending a POST request to the server, and function retrieving content from the opened tab. The GET function is responsible for retrieving the webpage in the form of the HTML code utilizing HTTP GET requests. While HTML parser retrieved from the currently opened webpage needs to be parsed, allowing only the relevant article text to be sent to the model. Meanwhile, the POST function sends the data to the server as an HTTP POST request and returns a response from the server, which indicates whether the data sent was assessed as fake or not.

To do so an attempt is made to find a <*article*> tag, and its textual contents then are read. The text is parsed so that all the tags like <*div*>, <*p*> or <*a*> as well as their attributes are removed. The text afterward is formatted, so that all whitespaces are deleted, and the text looks like an article genuinely written and is ready to be sent to the model.

Figure 1 presents the components of the overall system. The plugin acting as a client, allows the whole classification to take place in the server. The client component contains three elements: the browser extension, HTML sender an the receiver allowing the feedback from the server. The architecture focuses largely on the server side, including HTML receiver, data extractor, Fake News detector and decision sender. The data extractor, responsible for the feature extraction process, described above, produces the list of fake articles propagator. The classifier then, with the data extracted from the dataset using the feature process, permits for the final step on the detection of the news through the processes on the detector. Finally, the model was saved with *.tf* extension for later usage.

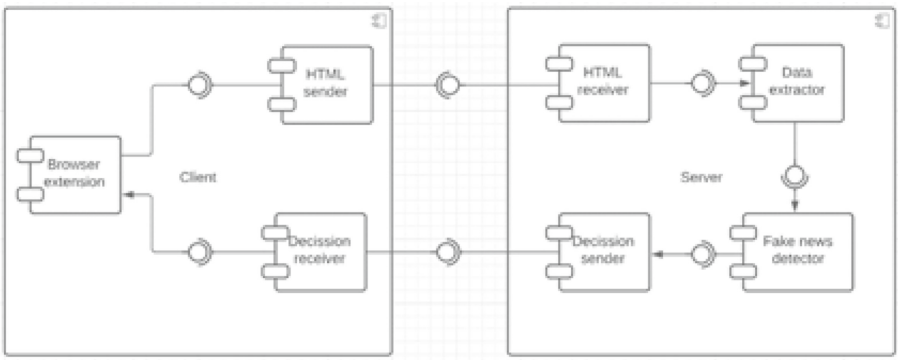

Fig. 1. System's architecture for fake detection.

4 Dataset and the Pre-processing of the Data

The whole system was tested on the primary dataset using a collection of news articles obtained from the open Machine Learning Repository Kaggle [11]. The data obtained from the Kaggle platform was split into two files: *True.csv* and *Fake.csv*. The dataset that is used is a public fake and real news dataset. The dataset is split into two parts: real news or fake news, almost perfectly balanced with around 20000 real and fake articles respectively. It includes features such as title, content, type of the article and publication date (Table 1).

Table 1. Data extraction features

No.	Attribute	Type	Description
1	Title	Text	Indicates the title of the Article
2	Content	Text	Indicates the content of the Article
3	Publication date	Date	Indicates the date the article was published
4	Label	Text	Indicates the Article labeled as Fake or True

The title indicates the title of the article. The content is the body of the article. The subject shows which type of news the article belongs to. Lastly, the date shows the publication date of the article. Considering that the network must analyse all the input text, the article should be split into smaller elements. Such an approach generally increases the speed of data processing and first and foremost, increases the accuracy of predictions [12]. The data is split into words mainly because such a technique is highly effective, considering that it is needed to trace and save the context for every word in the text (Table 2):

```
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
```

Table 2. Overview and comparison with the existing solutions

Solution	Approach used	Type output	Accuracy
FakerFact	*Deep Learning*	Assessment/Credibility score	66%
BRENDA	*Deep neural network*	Claim and User Feedback False/True	to 86%
TrustedNews [13]	*Machine Learning*	Objectivity score	73%
Check-it	*Deep Neural Network*	Classification Fake/True	to 90%

In the proposed solution, we use a special blacklist of words from the natural language toolkit (*nltk*) library, which makes it possible to exclude the prepositions and words without a high semantic load [13]. Having defined *fetch_data* function with the use of Pandas library it was possible to store data with dataframe objects. For the data pre-processing it was necessary to label positive samples and negative samples, then merge the title with the article content and drop irrelevant data:

```
def reorganize_data(true_news,fake_news):
  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,
    ran-dom_state = 1)
  return x_train,x_test,y_train,y_test
```

Positive and negative samples were merged into one dataframe object, and then split into training and validation sets. Functions provided by scikit-learn library allowed to shuffle the data and split with a given size of the validation set, namely 20%. To input, this data to the TensorFlow neural network, it is needed to get data converted to python *list_objects*. The main reason to use Neural Network as a text classifier is the huge flexibility and possibility to link with external architectures [15]:

```
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))
```

The model uses *nnlm-en-dim50*, a pre-trained neural network managing the embedding and tokenizing data. Except for the embedding layer, it contains one more hidden layer and an output layer with a *softmax* activation function (Fig. 2).

```
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
```

Fig. 2. The model of pre-trained neural network.

The model was trained with four epochs considering that a longer weight update was not necessary. For the visualization of the learning process, Tensorboard was employed, and after one epoch, the model achieved almost its maximum accuracy (Fig. 3).

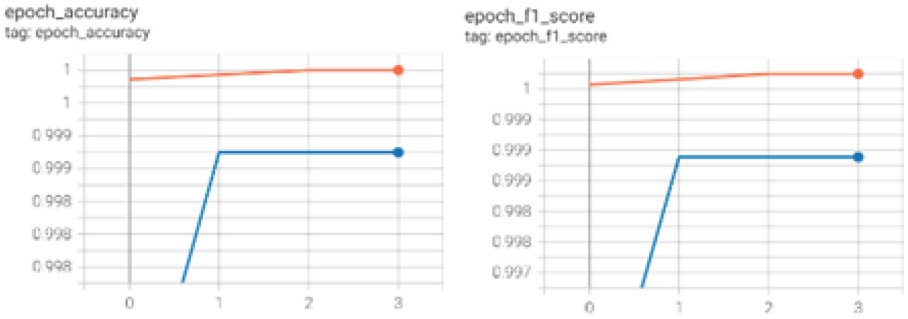

Fig. 3. Accuracy and F1-score of trained neural network model.

After three epochs the model reached significant results for the training and validation set (Table 3).

This signifies that clear separation can be observed between positive and negative samples and the architecture is adapted to the classification requirements.

Table 3. Accuracy and F1-score results of trained neural network model

	Training Set	Validation Set
Accuracy	100%	99.9%
F1-score	99.9%	99.9%

5 Cloud Deployment

The fake news detection system was deployed through the Google Cloud Function platform, mainly because of the advantages over other systems, namely functions that get triggered when an event is fired, hence terminated after execution of the function. Files such as *variables.index* and *variables.data-00000-of-00001*, were uploaded to Google Storage Bucket before the function was created and configured [15]. The first file stores the list of variable names. The second one stores the actual values of all the variables saved, and the HTTP allows unauthenticated invocations. The memory allocated by the function is set by four gigabytes due to the size of the second file on which the cloud function operates. For the source code of the cloud, there are two files, specifically *main.py* storing

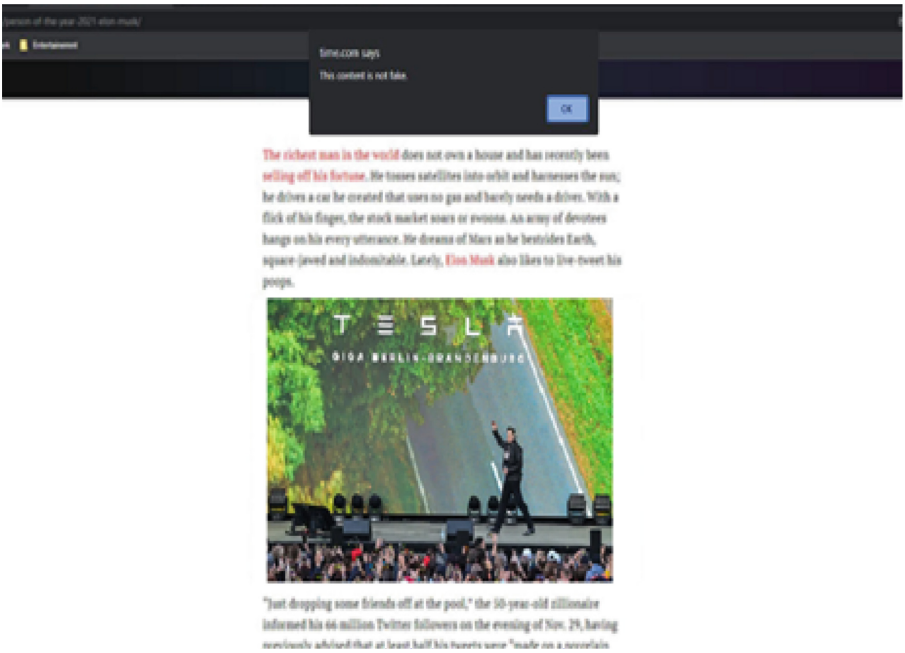

Fig. 4. The proposed extension in Google Chrome [15].

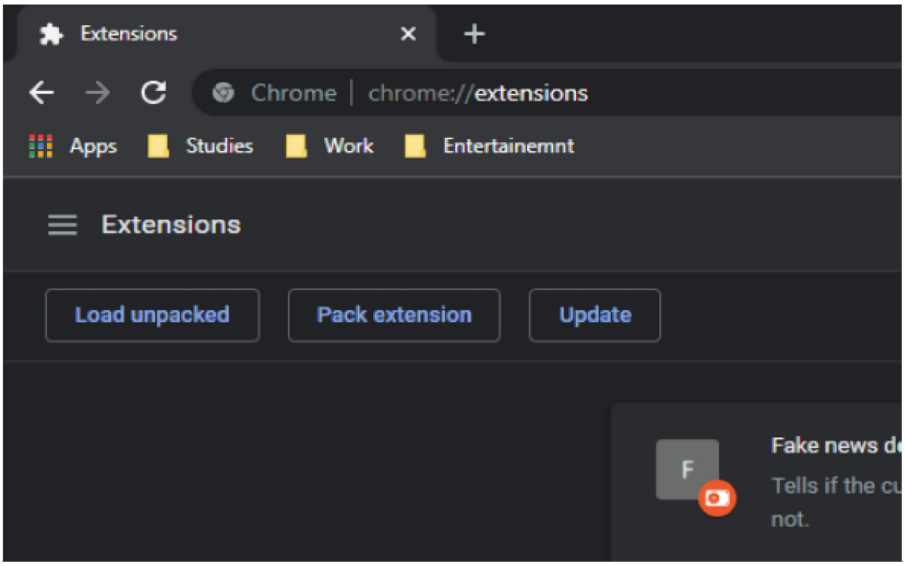

Fig. 5. Proposed extension panel in Google Chrome [15].

the function code and executing when the trigger event happens, and *requirements.txt*, where libraries required for function execution are declared (Figs. 4 and 5).

To retrieve the article content from the browser, the *is_fake* entry point of the cloud function is called, as a request to HTTP request for the model prediction. The function loads variables from cloud storage and stores them in a form of Blob objects [15]. Following, the variables are passed to the model, which is ready on prediction of the article veracity.

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* [16] results enable the final stage of cloud function action. Regarding the security, Google Cloud Platform provides a variety of security aspects for the Cloud Functions [17], in this case, the access control based on authentication. To allow all users the possibility of invoking fake news detection function, allowing this way also the full functionality, the access was set public. Furthermore, there is also a full control on permissions request possibilities, meaning agents and entities owe the control on cloud function resources access.

6 Discussion and Conclusions

Considering that the dataset contained articles revolving mainly US politics, in the project tests, there was a tendency to have articles with political content classified as fake. Nevertheless, this can be as a result of the fact that such content inflicts, in general, more controversy than any other topic or content.

The main advantage of the offered solution as elaborated in the sections above is the simplicity and the ease of use. It requires no more effort to verify the truthfulness of an article than enabling the plugin, with no need to access any other page, copy URL, or create an account.

There are, however, two main issues that require attention. First of which is the time required to get the result. Although it is very quick and easy to activate the extension itself, having a final output is not processed within the most optimal time. Considering that the existing solutions already offer output within 5 s, it is still not the best user-friendly option. The reason for that is the fact that in the cloud function the model gets downloaded each time the function is called.

The other issue to highlight is the fact that times the HTML cannot be processed even though the site is an actual article. Various sites are built differently when it comes to the 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, it is still possible to encounter ones with a different structure. Hence, with the implementation of parsing the contents of <*article*> tag, the article cannot be extracted and formatted. It certainly is not common on well-established sites.

Regarding perspectives and future work, some aspects might need further improvement, that is user interface, article recognition, and execution time. The

first aspect concerns the way how the browser extension is presented to the user. The interface might be developed further to be more appealing. Additionally, the other aspect regards the fact that the software does not work for websites that do not include an *<article>* tag in their HTML structure. Therefore, the extraction of content shall be added despite the structure. Lastly, the aspect of processing time, to improve that, the model loading needs improvement and re-organization. Enhancing the features mentioned above would lead to better performance and significantly improved version even in comparison to the state of art solutions.

Acknowledgment. The publication was created as part of the participation in the project of Polish National Agency for Academic Exchange under the "STER" Programme – Internationalisation of Doctoral Schools" as part of the project "Curriculum for advanced doctoral education & training – CADET Academy of Lodz University of Technology".

References

1. Paschalides, D., et al.: Check-it: a plugin for detecting and reducing the spread of fake news and misinformation on the web. In: 2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI), pp. 298–302 (2019)
2. Alonso-Bartolome, S., Segura-Bedmar, I.; Multimodal fake news detection. arXiv:2112.04831 [cs] (2021). Accessed 16 Feb 2022
3. Watson, A.: Fake news in the U.S. - statistics & facts, Statista (2022). https://www.statista.com/topics/3251/fake-news/. Accessed 02 Mar 2022
4. Choudhary, M., Jha, S., Prashant, Saxena, D., Singh, A.K.: A review of fake news detection methods using machine learning. In: 2021 2nd International Conference for Emerging Technology (INCET), pp. 1–5. (2021). https://doi.org/10.1109/INCET51464.2021.9456299
5. Khanam, Z., Alwasel, B.N., Sirafi, H., Rashid, M.; Fake news detection using machine learning approaches. In: IOP Conference Series: Materials Science and Engineering, vol. 1099, no. 1, p. 012040 (2021). https://doi.org/10.1088/1757-899X/1099/1/012040
6. Kula, S., Choraś, M., Kozik, R., Ksieniewicz, P., Woźniak, M.: Sentiment analysis for fake news detection by means of neural networks. In: Krzhizhanovskaya, V.V., et al. (eds.) ICCS 2020. LNCS, vol. 12140, pp. 653–666. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-50423-6_49
7. FakerFact. https://www.rand.org/research/projects/truth-decay/fighting-disinformation/search/items/fakerfact.html. Accessed 16 Feb 2022
8. Botnevik, B., Sakariassen, E., Setty, V.: BRENDA: browser extension for fake news detection. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Etrieval, New York, NY, USA, pp. 2117–2120. Association for Computing Machinery (2020). https://doi.org/10.1145/3397271.3401396
9. Trusted News. https://trusted-news.com/. Accessed 16 Feb 2022
10. The Factual - Unbiased News, Trending Topics - The Factual. https://www.thefactual.com/?lp=new. Accessed 16 Feb 2022
11. Fake News—Kaggle. https://www.kaggle.com/c/fake-news/data. Accessed 11 Feb 2022

12. Abdulrahman, A., Baykara, M.: Fake news detection using machine learning and deep learning algorithms. In: 2020 International Conference on Advanced Science and Engineering (ICOASE), pp. 18–23 (2020). https://doi.org/10.1109/ICOASE51841.2020.9436605
13. Kevin, V., et al.: Information nutrition labels: a plugin for online news evaluation. In: Proceedings of the First Workshop on Fact Extraction and VERification (FEVER), Brussels, Belgium, pp. 28–33. Association for Computational Linguistics (2018)
14. Hasimi, L., Poniszewska-Maranda, A.: Ensemble learning-based fake news and disinformation detection system. In: 2021 IEEE International Conference on Services Computing (SCC), pp. 145–153 (2021). https://doi.org/10.1109/SCC53864.2021.00027
15. Abiodun, O.I., Jantan, A., Omolara, A.E., Dada, K.V., Mohamed, N.A., Arshad, H.: State-of-the-art in artificial neural network applications: a survey. Heliyon **4**(11), e00938 (2018). https://doi.org/10.1016/j.heliyon.2018.e00938
16. Czyzewski, A., Lech, E., Milosz, A., Kowalski, K.: Cloud computing system - Raport. Lodz University of Technology, Student paper (2022)
17. Multi-Class Neural Networks: Softmax—Machine Learning Crash Course, Google Developers. https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax. Accessed 16 Feb 2022
18. Securing Cloud Functions—Cloud Functions Documentation, Google Cloud. https://cloud.google.com/functions/docs/securing. Accessed 16 Feb 2022