|  |  |
| --- | --- |
| Trust  It |  |
|  |  |
| 5.2.2020  Authored by: Natalija Pesic  Mentor: Manuel Garcia Vega |  |

**Table of Contents**

[Bibliography and tools 3](#__RefHeading___Toc1521_3852955197)

[An example of one scraping script: 4](#__RefHeading___Toc1803_3852955197)

[Sources for training NN 5](#__RefHeading___Toc2243_3852955197)

[Connecting back-end and front-end 5](#__RefHeading___Toc1807_3852955197)

[Time spent working on the project 6](#__RefHeading___Toc1523_3852955197)

[Description of the practice 7](#__RefHeading___Toc209_2878161577)

[Building a database 7](#__RefHeading___Toc211_2878161577)

[Algorithm for evaluating the news 8](#__RefHeading___Toc213_2878161577)

[Algorithm that uses the database, google search and NN 10](#__RefHeading___Toc369_1704329890)

[List of the problems raised and the procedure followed for resolution 12](#__RefHeading___Toc215_2878161577)

[Identification of the contributions that, in terms of learning, have involved practices 13](#__RefHeading___Toc217_2878161577)

[User interface 14](#__RefHeading___Toc229_2878161577)

[The form 14](#__RefHeading___Toc231_2878161577)

[The slides 14](#__RefHeading___Toc233_2878161577)

[Evaluation of practice and suggestions for improvement 16](#__RefHeading___Toc219_2878161577)

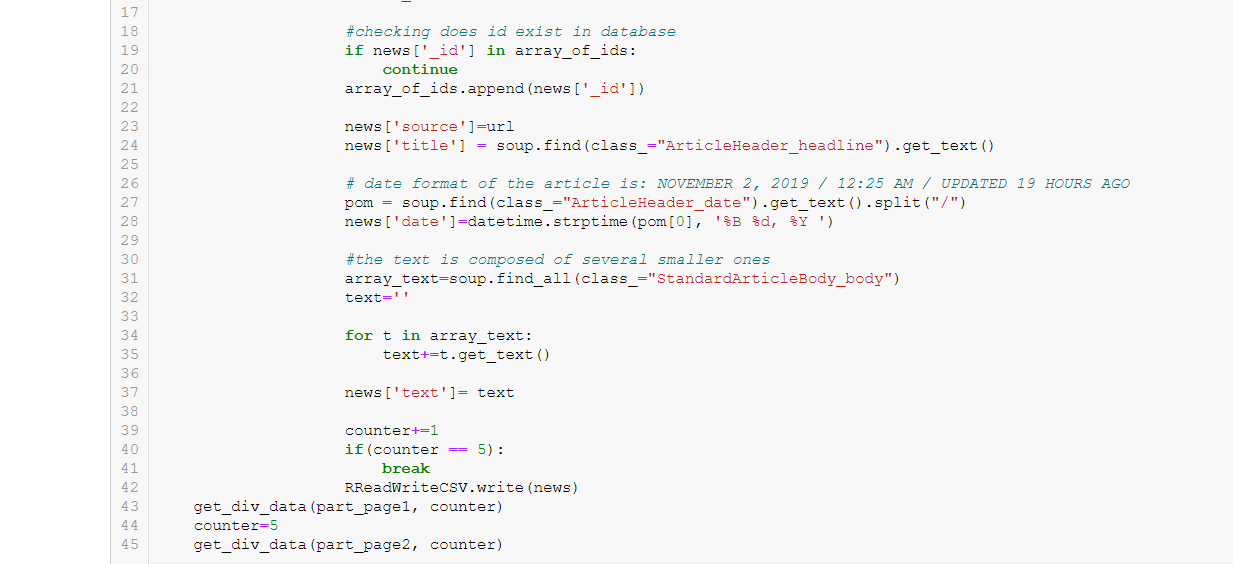
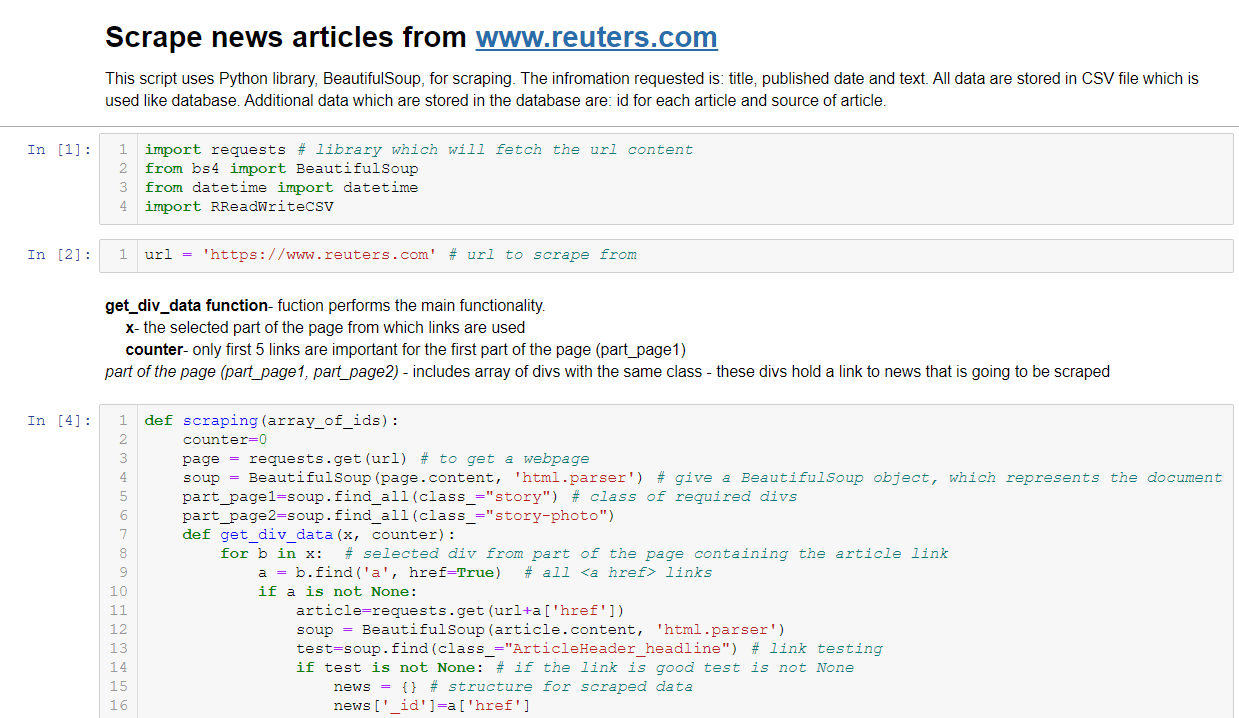
# **Bibliography and tools**

After research and founding out the list of reliable news, I choose some of them. Here is a list of chosen:

**Sources that were used like reliable news are:**   
*Al Jazeera (*[*https://www.aljazeera.com/*](https://www.aljazeera.com/) *)  
Associated Press (*[*https://www.ap.org/en-gb/*](https://www.ap.org/en-gb/) *)  
BBC News (*[*https://www.bbc.com/news*](https://www.bbc.com/news) *)  
Bellingcat (*[*https://www.bellingcat.com/*](https://www.bellingcat.com/) *)  
BuzzFeed News (*[*https://www.buzzfeednews.com/*](https://www.buzzfeednews.com/) *)  
Deadline News (*[*https://deadline.com/*](https://deadline.com/) *)  
Desert News (*[*https://www.deseret.com/*](https://www.deseret.com/) *)  
The Economist (*[*https://www.economist.com/*](https://www.economist.com/) *)  
The New York Times (*[*https://www.nytimes.com/*](https://www.nytimes.com/) *)  
PolitiFact (*[*https://www.politifact.com/*](https://www.politifact.com/) *)  
Reuters (*[*https://www.reuters.com/*](https://www.reuters.com/) *)  
Snopes (*[*https://www.snopes.com/*](https://www.snopes.com/) *)*  
*The Atlantic (*[*https://www.theatlantic.com/world/*](https://www.theatlantic.com/world/) *)*  
*The Hill (*[*https://thehill.com/*](https://thehill.com/) *)*  
*Vox (*[*https://www.vox.com/*](https://www.vox.com/) *)*

For scraping those news were used programming language Python and library Beautiful Soup.

### An example of one scraping script:

After scraping, data are stored in database in CSV format.

### Sources for training NN

Sources, that were used for training neural network are store in database namely two CSV files, one for reliable news and another for unreliable news. And it makes one type of input in neural network. Neural network model used is Multi Layer Perception imported from “scikit learn” python library.

Another source or input in database is Google results after searching for specific title.  
For scraping Google results were used SerpWow, real-time SERP API.

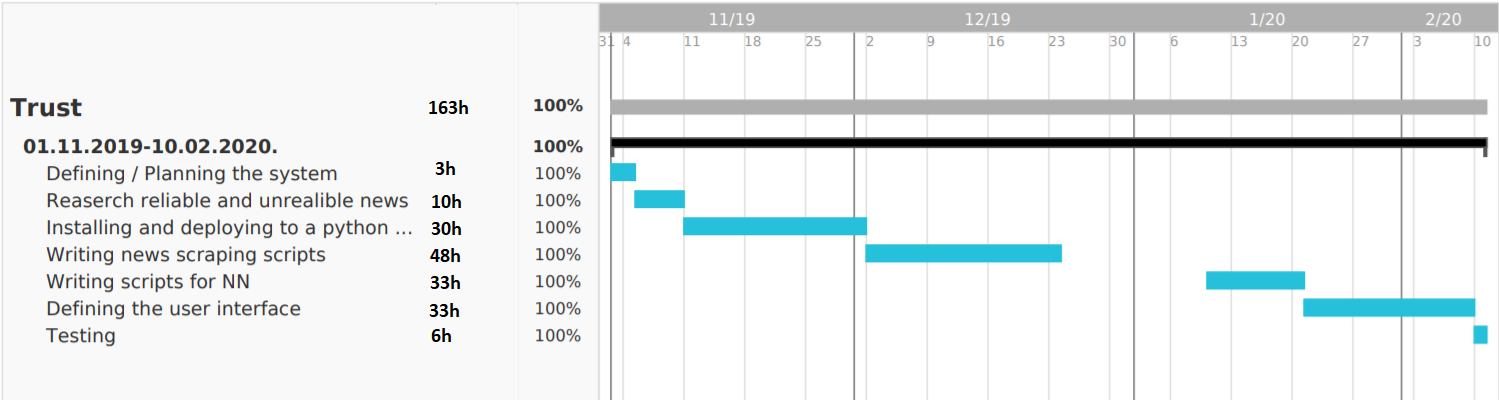
### Connecting back-end and front-end

On the back-end side was used Python and for the front-end side html, css and js. Django web framework is used to make connection between them.

# Time spent working on the project

# 

# 



The project is divided in 7 phases. On the chart is shown name of the phase and day spent work on it (weekend aren’t included). In general, I was working 3 hour per day except in second and third phase.

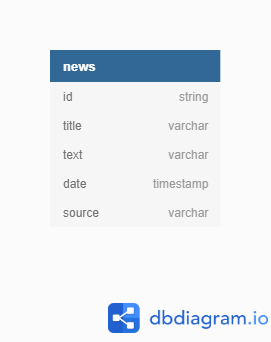
# Description of the practice

Our goal was to make an application that could detect a fake news and, in that manner, help the users distinguish lie from truth. The making of an application consisted from 3 phases:

1. building a database of news for which we could claim it is reliable or not,
2. making an algorithm that uses the database, google search results and a neural network and
3. making an interface for the users

## Building a database

Our database consists of two parts: reliable and unreliable news. At first, we used MongoDB for holding our data but because our database has only one table (image 1) we decided later to use a csv files for storing the data.



*Image 1*

The reliable and unreliable news are saved into different files, so the attribute on weather a news is unreliable is unnecessary.

My part was to get the reliable news into the database. The sources considered as such are *Al Jazeera, Associated Press, BBC News, Bellingcat, BuzzFeed News, Deadline News, Desert News, The Economist, The New York Times, PolitiFact, Reuters, Snopes*, *The Atlantic*, *The Hill*, *Vox.*

The news is inserted from internet into the database using python scripts. Each python script scrapes news from one source. The main script calls a scraping function from each python script, so once it is running, news from all the sources are scraped and written into the database.

The *ReadWriteCSV* script handles reading and writing in csv file. Once all the news are scraped and loaded into an array of program, they are written in the file all at once using a *dump* function.

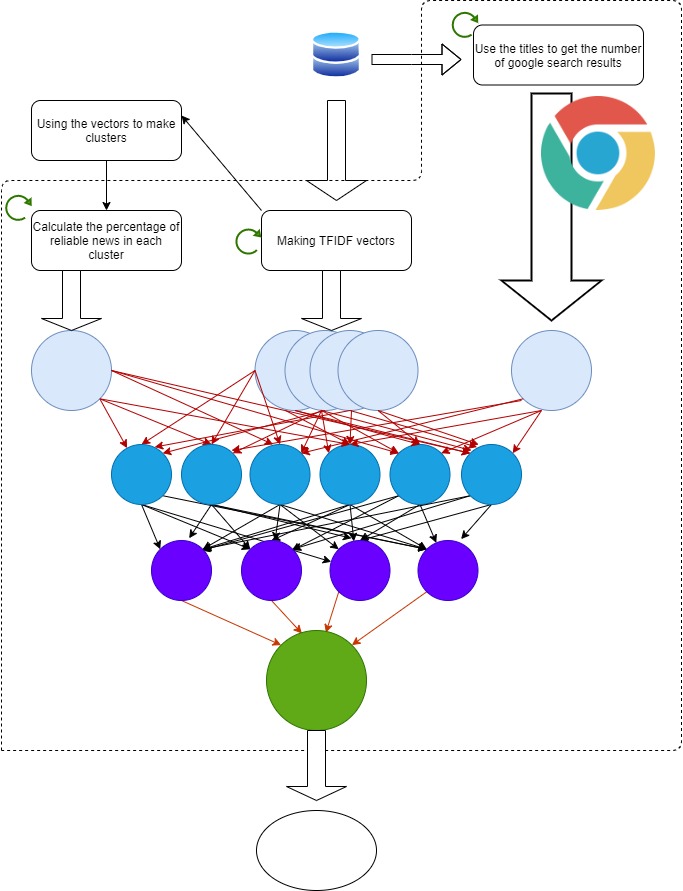
## Algorithm for evaluating the news

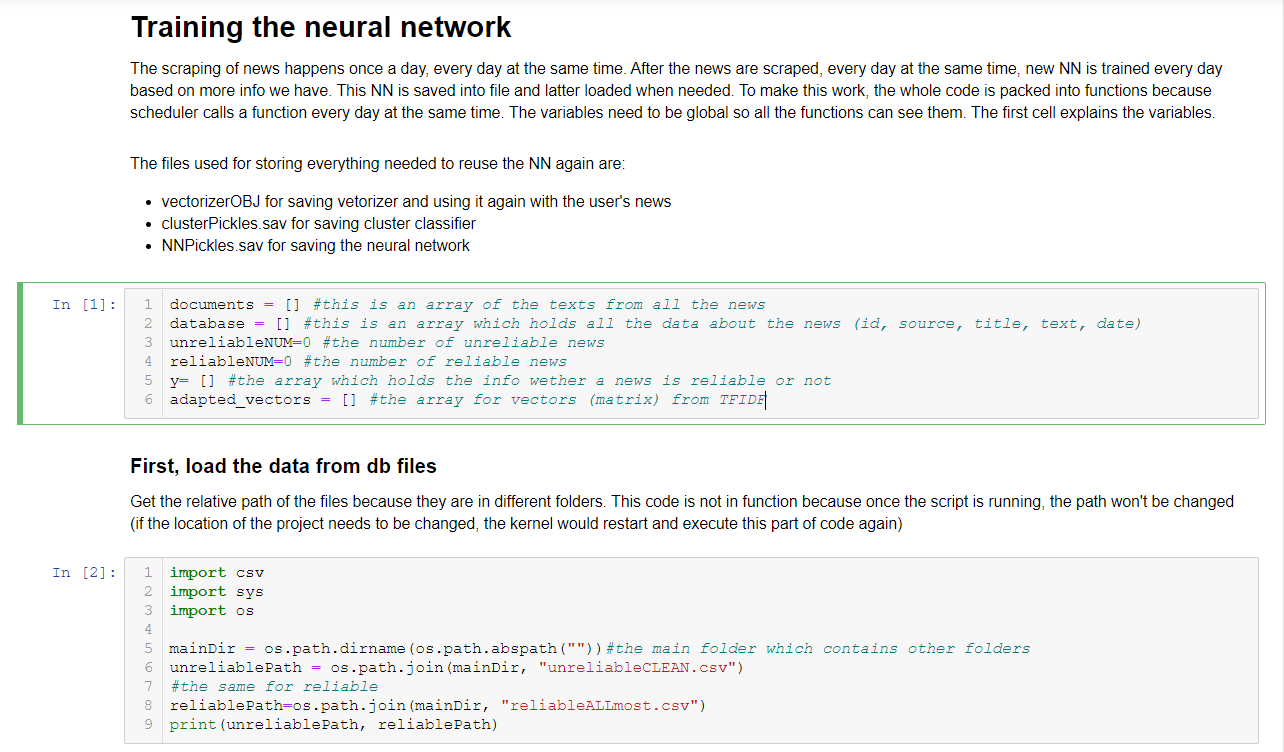
For making a prediction on weather a news is unreliable or not, we use two factors: the database and the google search results.

The algorithm first loads all the news and calculates the TFIDF (Term Frequency Inverse Data Frequency) vectors. These vectors are one part of input for neural network. Another part is the number of Google search results for the title of the news. With this data, a neural network (multilayer perception) is trained. Both TFIDF vectorizer and neural network model are saved in files (pickled) so they could be used again later.

When a query is represented for evaluation, a vector is made from this text using saved TFIDF vectorizer. Also, a number of Google search results for the query is obtained.

Finally, using **the vector made up of the words** and **number of research results on Google for the query** as inputs for neural network, a final conclusion is made on whether a news can be considered as true or not.





## Algorithm that uses the database, google search and NN

The interface is a web application written with html, js and css. The user should enter the date, title and text of the news for which he wants the evaluation. Then, after pressing a button, the data entered is sent to the server using fetch method. Server then evaluates the news using the already described algorithm and sends the evaluation back to front-end in JSON format. The result is rendered.

The result is described with status *reliable* or *unreliable.*

On the web page, the user can also see the news recently scraped from reliable news sources and can visit the source of those news.

# List of the problems raised and the procedure followed for resolution

The problems that occurred with the projects are mostly in relation with encountering with, for us, new technologies. The most time we spent was for learning how to work with python, CSV files or connecting python and front-end web page. First, we chose to work with MongoDB because database already had efficient mechanisms for checking key value and writing. Later, when we realized that our database is just one entity and that we could manage it on our own, we transited to CSV file. Because of this we had to change all of our scripts and to test everything again and of course new bugs appeared which required extra time to solve. To check if the key already exists, at the beginning of the scraping process all ids from the file are loaded into an array and then while scraping, new ids are compared with those from the array and, if for a news is decided to be in a database, its id is also added in the array of ids. As for writing, first, we tried to write in the CSV file each time a news is scraped. That meant opening and closing the file each time, which is around 300 times in total per day. This required a tremendous amount of CPU time. That is why writing in the database file now consists of two parts: call of *write* function which just adds a news object in a global array and at the end of the scraping all the news from all the sources a call of *dump* function writes a series of objects from the array into the database file.

All of these problems weren’t difficult to solve, but they consumed a lot of our time.

In conclusion, I can say that small mistakes at the beginning of the project and bad decisions in the first phase make big problems later in the development.

# Identification of the contributions that, in terms of learning, have involved practices

While building a database in MongoDB, learning how to use this database was for us not so difficult since we already had the knowledge from database subjects, although we had more experience with standard SQL bases and this is a NoSQL database. Later, when making a CSV file for database, our mechanisms for management of writing and reading were based on what we learned how regular database functions work.

For writing the algorithm that evaluates the news, the knowledge obtained from other subjects like *Data Mining* helped us to choose the best scikit-learn classifications (Multilayer Perception for neural network) and how to use them. Basic knowledge from general subjects like Mathematics helped us understand vectors in multidimensional space and how to use them.

Since we already made web applications within our web subjects, making interface was not a big problem. We never used python as back-end but we learned quickly what to use to connect the front- and back-end.

Organization of the teamwork was made with experience from previous subjects that required team projects and with theoretical knowledge from subjects about the management of large groups.

# **User interface**

The interface is a web application written with html, js and css which is connected with scripts using Django framework.

# 

The main page which can see above is made up of the title, form and slides.

### The form

The user should insert the text of the news whose he interested in. After clicking on the button “SEE RESULT..” a status of the news, for whose the text was inputted, will be shown.   
*The picture above is showing state of the page after button click.*

### The slides

This part of the page show some reliable news whose have been stored in database.  
Swiping through news user can perform by clicking on the buttons “**>”** and “**<**”.  
The button “READ MORE” opens a new tab with specific news whose title has been shown on the slide.

# Evaluation of practice and suggestions for improvement

The practice was very interesting. The research both on a technical part of realization of the application and on choosing sources of news, was informative, refreshing and of course will be useful for future work.

The experience of working in a different country and speaking on another language then our own teaches us to be flexible and how to work under various circumstances.

The topic is from a Natural Language Processing subfield opened for us new horizons in a way that we would like to continue the research in that direction.

We have two suggestions for improvement:

The firs would be a more specific classification of the news. The news should be categorized in more groups like false, true but with some false interpretations, pseudo-scientific, opinions, satirical and news composed from tweets and quotes. Granulating the classification would give more precise data and hence the prediction would be more precise.

The second suggestion is to use different, more soothing machine learning models special for NLP. We don’t have more knowledge on those so we used usual algorithms.