Fake news detection in real time

Technical report

Project created by:

* Munteanu Andrei-Ștefan
* Răzvan Baisan
* Crîșnuţã Iulian-Gabriel
* Victor Vlad
* Pantelemon Victor-Stefan

Under guidance of professor:

* PhD. Adrian Iftene

# The problem.

The problem consists into problems which are visible every day when we are navigating on twitter. That problem is that we deal with lots of fake news and fakes user. Our scope with this project is to determine the probability of a user to be fake or not.

Another problem would be that there aren’t almost any real time detection tools which could do this for a home user which is browsing the web. Most of the tools on the web need an insane amount of time to do semantic analysis over that topic or to analyze the comments, reactions and other types of forms a user has over a unique post. This kind of tools aren’t actually a good tool for fake news which are starting from multiple sources at the same time making a web explosion of fake news over twitter especially.

# Functional requirements for our solution

All information will be collected from twitter via api (posts, user information (number of posts, followers, account creation date, etc.). We will also have user profiles built (characterized by details about the user, user activity, reactions of other users to what he does on the network, etc.) All this information will be used in order to have:

* A metric will be proposed to calculate the credibility of a post
* A metric will be proposed for credibility of a user

All the available metainformation should be taken into consideration. Also, we should have a list of fake users centralized in order to get a better response, after the front-end is putting the user in a category. Resources should be centralized in order to get faster responses for already processed news. Resources will be used outside the Twitter network to validate the information (Google, blogs, newspapers, etc.) via a crawler in order to see if the information from the post/user and their metric can be considered trustful. Fronted will be able to show extra information about metrics also it should be able to edit the strictness level for marking news with the possibility of being fake or not. Fronted will show analyzed resources. User will also be able to mark a post as fake manually.

# Our solution

Our fronted is made via a chrome extension which allow us to add functionality to the Chrome web browser without diving deeply into native code. Fake news browser extension was built with core technologies that web developers are very familiar with - HTML, CSS, and JavaScript. The only thing different from classic web pages are functionality to Chrome through some of the JavaScript APIs that Chrome exposes. Fake news extension was also created to work only on certain pages through the use of Page Actions, it can run code in the background using Background Pages, and can even modify an existing page loaded in the browser using Content Scripts. Also, a manifest.json file was created to tell Chrome important information about extension, like its name and which permissions it needs. Also, the extension is made of different, but cohesive, components. Components include background scripts, content scripts, UI elements and various logic files.

Our backend was composed via multiple tentatives and we tried to include all classes from advanced software engineering techniques into it, so it may be considered our playground. We have started with the methodology being a model driven development one. Then with the architecture n tier pattern which was done with model first entity framework and database first with entity framework core in order to stay in our methodology grounds. The backend is a restful api which respects all SOLID principles and it is secured via authentication and authorization having multiple keys and rights. It is also secured having a token for each user generated. The deployment is done into Azure cloud due the fact that is a very friendly environment which supports also CI/CD in order for us to be able to build and deploy faster. Also, we were thinking to go from GitHub to the VSTS platform due the fact that is easier to follow all tasks assigned with repos and with pipelines for CI/CD and test plans. Our architecture is service oriented and we use in our project python, JavaScript and C#.

The crawling service is a simple python script which is able to crawl the google with the scope of finding articles which could contain interest keys for us. Then those links are saved and they are opened in parallel in order to crawl them as well and see if that information is or not of any help in our process unit.

The scrapper for twitter (\* requires developer account for API keys, \* runs non-stop by searching for certain keywords) is able to save data in mongo collections about user, text, date of post, number of likes, retweets, if it was posted as a response, but also data about user. It’s exporting from mongo a json which is then used in a Redis queue with the scope of being processed with multithreading. On its part we are also using a Bayes and other ML algorithms to see if the tags like emails or other things match with our keywords needed. Also, this API allows us to give a credibility log. Score over the users in terms of fakeness.

Process Unit is a server that receives requests with twitter URL Based on a tweet, we can extract the text content and query Google with the tokenized (cleaned up) text. We find articles from which we extract the body and compare semantically with the tweet. We average with all the results found and that's the score.

Some important things that didn’t come along as expected with this module it would be that:

* it did not come out to extract semantic relationships from the text. It is possible to classify words into speech / sentence parts but we couldn’t do it in time to use pairs to check ontologies.
* We tried instead to compare the whole article with the tweet, to take each sentence and compare it, but it takes a long time after that. We could use a heuristic with which to exclude sentences that we are not interested in to, at least that was our final resolution in order to make it faster.

In order to implement this, we used modules for:

* Tokenization, lemmatization, stop words – nltk
* Semantic analyze – genism and nltk
* Tweepy – twitter api
* Scrapping – google search
* Extract information from articles – newspaper
* Calculate things – numpy
* Server for module – flask

All our presented modules are designed with aspect-oriented programming and monitoring oriented programming in mind, also for critical paths we developed functional testing (unit testing) and from non functional perspective we created a collection in postman. In order to have the project in the current form we did a lot of refactoring. Now almost everything is decupled and can be replaced with anything.

Comparing our solution with others it isn’t the best from our minimal tests till now, from the point of metrics resulted not architecture, an attached annexes which will come will describe this study with more statistics, but from our tests we can only get high confidence of a fake news only when we have some more information about it and the text is on other safe sources already.

# Future work

We identified three main objectives which can be pursued in order to greatly improve the overall robustness of our project:

* Sentence analysis
* it would make it easier for us to exclude sentences that are not strongly related to the subject tweet (either heuristics or something else)
* analyzing individual question would make it easier to detect negations (which we currently don't detect)
* Semantic relation extraction and evaluation
* useful for occasions when the tweet content is simple and we cannot rely on written articles about the topic
* Distributed computing for demanding steps
* some steps can benefit from distributed computing, such as web scraping and sentence analysis

# Conclusions

Having fake news detection as primary objective, our proposed method can have decent results when focused on tweets about popular topics that can be verified via written articles found on the web.

The method is not suited for tweets that make false claims such as *"Europe is an ocean"*. For this kind of task, we reckon that semantic relation extraction and evaluation using an ontology is a far better way to determine the truthiness of such claims.

# Bibliography / Links

Semantic Fake News Detection: A Machine Learning Perspective  
https://www.researchgate.net/publication/333627738\_Semantic\_Fake\_News\_Detection\_A\_Machine\_Learning\_Perspective

The language gives it away: How an algorithm can help us detect fake news  
http://theconversation.com/the-language-gives-it-away-how-an-algorithm-can-help-us-detect-fake-news-120199

Case study: Use of Semantic Web Technologies on the BBC Web Sites  
https://www.bbc.co.uk/blogs/bbcinternet/2010/02/case\_study\_use\_of\_semantic\_web.html

Personalized Semantic News: Combining Semantics and Television  
https://www.researchgate.net/publication/221435652\_Personalized\_Semantic\_News\_Combining\_Semantics\_and\_Television

Linked data and the semantic web  
https://www.bbc.co.uk/academy/en/articles/art20130724121658626

Detecting Fake News in Social Media Networks  
https://www.sciencedirect.com/science/article/pii/S1877050918318210

Accessing the Twitter API with Python  
https://stackabuse.com/accessing-the-twitter-api-with-python/

The Language of Fake News: Opening the Black-Box of Deep Learning Based Detectors  
https://cbmm.mit.edu/sites/default/files/publications/fake-news-paper-NIPS.pdf

Fake News Detection on Social Media: A Data Mining Perspective  
https://www.kdd.org/exploration\_files/19-1-Article2.pdf