CentraleSupélec

Fake News detection through virality analysis on social media

- Research Project -

Authors :
M. Thibaut Hurson
M. Mathurin Maltot

Teacher :
M. Richard Combes

Contents

1 Introduction

Since the 2012 American elections, fake news have grown in both popularity and influence, becoming a major threat to our democracies. Spread widely on social media, they break the trust of voters in news stories and play an active role in the polarisation of our society. In an effort to curb their impact, social media are trying to detect such news and stop their spread. Given the massive amount of news, manual verification would be too costly. Thus, they rely on classification algorithms to parse the networks and look for fake news. In this study, we are going to focus on the network-based approaches that are less popular yet very promising. Leveraging data such as reach, user interaction or propagation speed we are going to build a machine learning model to predict the veracity of a given information.

2 Our Approach

Traditionally, there are 3 main ways to do Fake News detection : linguistic-based, user-based and propagation based. The first one relies on NLP (Natural Langage Processing) and is the most popular. The second one relies on scoring mechanism. In our study, we have chosen to explore the third one, which is less known than the other two. This choice was based on an article that showed clear differences on the propagation of news with regards to their veracity ([14]). We then came up with the idea of using supervised learning to exploit theses differences and build a classifier to detect and predict the veracity of such news.

We identified three main steps for our project. The first one would be to get a dataset. Actually we needed to build it from scratch (See "Main Difficulties" for more details). On a second hand, we would need to extract the chosen features for each news story. Lastly, once we have selected the features for our model, we would need to build it, train it and validate its accuracy.

3 Selecting a Social Network and building a dataset

First, we needed to choose which Social Media to work on, which data entries to select and build our entire methodology.

3.1 Choosing a social media

When selecting a social media, we looked at the pros and cons of choosing a particular social media. We chose to work on Twitter. Twitter is a microblogging and social networking service on which users post messages known as "tweets". Such tweets can they be "liked", "commented" or "retweeted". This has some key advantages regarding our study. First, the news propagation

is more streamlined than in other social media: an user post a tweet and other users retweet it and so on. This makes for tree-like structure that are simple to study: cascades. Other social media such as Facebook are thougher to analyse : the "Share" feature is not as used, comments are more important and the network is structured in quite distinct groups. On Twitter, as long as you follow an individual, you are exposed to its content and the reactions of other users on a particular topic. Moreover the Twitter API made for an easy way to scrap tweets and construct the database (at first).

3.2 Scrapping Twitter

In order to scrap data from Twitter, our first idea was to use Twitter API. However we quickly understood that this API was putting limits on the number of tweets that we could retrieve per amount of time (We were limited to 150 API requests per 15min, even with a developer account). Thus, we searched for an alternative and found the Twint library. Twint is an advanced Twitter scraping tool written in Python that allows for scraping Tweets from Twitter profiles without using Twitter's API. As Twint pulls tweets that contain particular words, we try to select entries that are narrow enough to define one unique topic and we get all tweets and retweets related to it. Then, to build a cascade, we manually select a tweet and we implemented algorithms to link this tweet to its retweets. The detailed process will be explained in the following section.

The dataset was built around fact-checked rumors by organizations such as snopes.com, politifact.com, or factcheck.org.

3.3 Building a Cascade

In order to build a cascade, we need to understand how data is scrapped from Twitter. We have two types of data: Tweet and Retweet. Each Retweet is associated to the initial Tweet, which means that we do not have the information about the propagation of the tweet between different users. For example, let us consider an original tweet a, a retweet b of a, and a retweet c of the retweet b. Then, with the raw data we only have the information that b and c are retweets of a.

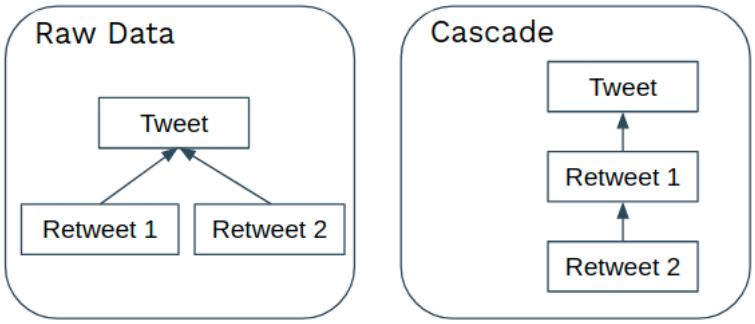

Figure 1: Raw Data versus Cascade

In order to get the propagation information, the procedure works as follow. For each initial tweet, we consider its retweets. Then, we assume that among the retwitters, those that are following the original twitter are direct retwitters of the original tweet. Then, for each direct retwitters, we consider that retwitters that are following the direct retwitter and not the initial one are retwitters of the direct retwitter. And we continue this procedure until the cascade is created. During this process, we assume that if a retwitter c follows for example two other retwitters a and b, then c has retwitted the one that has retweeted first.

By running this algorithm on the raw data obtained from Twitter with the twint library, we are able to build cascades like the ones shown below. Please find below 3 cascades, representing the propagation of 2 fake news (Figure 2 and Figure 4) and 1 true news (Figure 3).

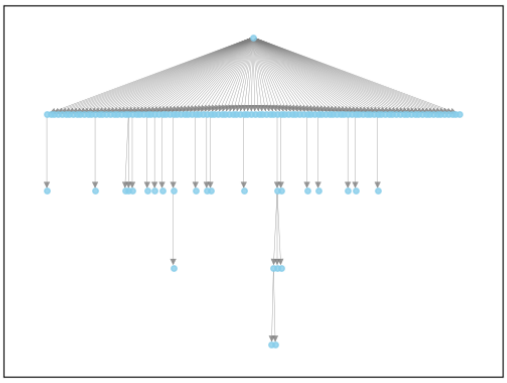

Figure 2: Fake News Cascade

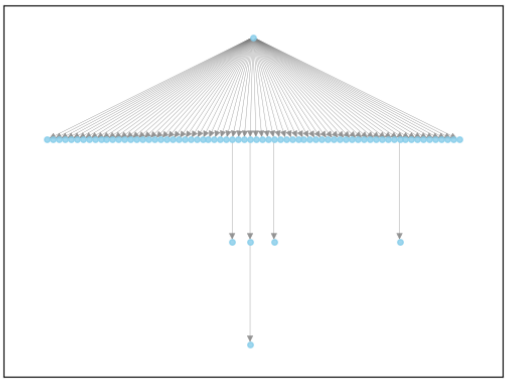

Figure 3: True News Cascade

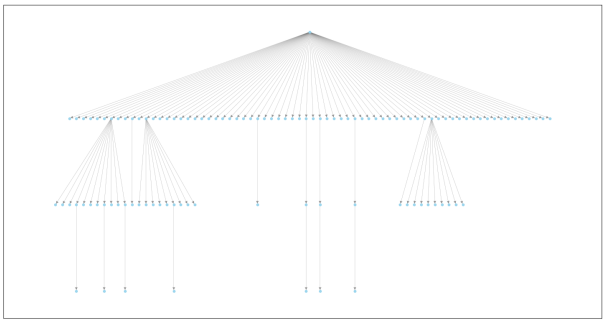

Figure 4: Another example of fake news propagation

The fake news cascade displayed above are from two different news. The first one, was a news about a reality show in which an actress was accused of witchcraft (Figure 2). The second one was on the fake snow conspiracy theory to explain snowstorms in Texas that occurred in March 2021 (Figure 4). The true news selected here was about the launch of the season four of Shingeki no kyojin, a japanese anime (Figure 3).

3.4 Results and limits

Using the aforementioned technique, we were able to generate a certain number of news propagation graphs. However due to the fact that to test follower-relationship among retwitters we need to use Twitter API, there was a limit on the number of requests we could make. As this limit is fixed to 150 requests every 15 minutes, we could only generate small propagation

graphs. For example, for a tweet that was retweeted 150 times, we could generate its propagation graph after one day. At first, we tried running algorithms over night but even with this strategy, we were getting propagation graph in a too small rate. Moreover we estimate that we would need to build propagation graph with at least 200 retweets to be able to analyze graphs and find insights on the differences between true and fake news.

4 Simulating propagation

As a first step to model the propagation of news on social media, we decided to simulate what could happen on an fictitious graph. This approach allowed us to emulate the behavior of a Twitter sub-network, without relying on the twitter API.

4.1 The Network Model

In order to get an accurate representation of the real life situation, the simulated network must have similar properties as real social network.

According to literature, this is characterized mainly by 4 properties :
- Power-law (heavy-tailed) degree distribution : Many nodes with only a few links and a few hubs with a large number of links.
- Small world property : the average distance between two node should be relatively small
- Large clustering coefficient : high number of triangles in the network (a friend of a friend has a high probability of also being a friend)
- Giant connected component : No big community is fully excluded from the rest of the network

Initially we though about using Erdos-Renyi graphs. But they didn't present all of the properties we were looking for. In particular, the degree distribution didn't match the power law, but rather a Poisson one.

In order to fix the degree distribution issue, we implemented some configuration model. This allowed us to get a correct power-law degree distribution, but it lacked the small world aspect that is so important in real life network.

A 3rd type of graph we looked at is the Barabási–Albert graph, which is also scale free and has also a giant connected component. This kind of random graph was particularly interesting for our study, with the existence of big Hubs (nodes of high degree) that exists also in real life networks. Actually, the Barabási–Albert model uses preferential attachment : a new node coming to the network has a high probability of connecting to the most popular one. Evidence in practice on Twitter showed that this was not always true as people formed communities, hence having high clustering coefficient.

Finally, to take into account the "small-world" phenomena of real life graph, we ended up choosing to use Watts–Strogatz graphs. Indeed, even if we loose the degree distribution property, those graphs are well connected, while having a small average distance and large clustering coefficient.

Please find below a comparison of the different graph models :

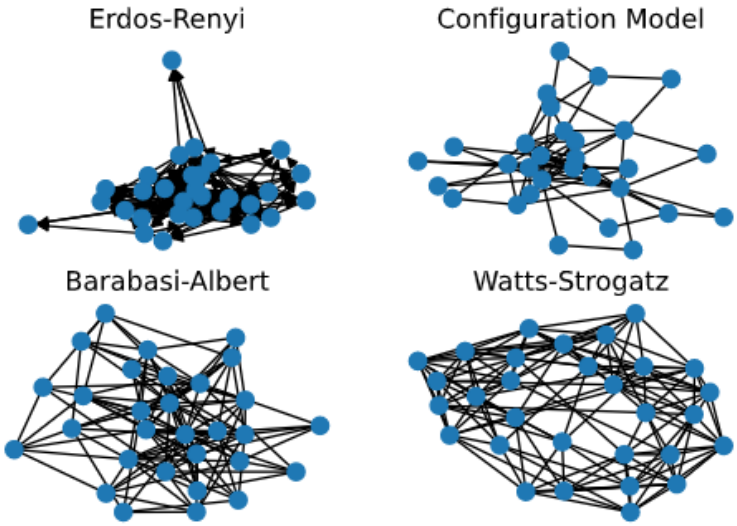

Figure 5: Different Graph Models

This generated graph is going to be useful for testing our hypothesis and model in a more controlled environment before using real-life data.

4.2 The Propagation Model

Once we built the network model, it stays constant throughout the propagation study. We then had to set up a propagation framework to generate the cascades.

We started with a quite simple algorithm :
- Init : The news reaches the network at a random node
- At each step : The news has probability p_{true} (resp. p_{false}) to spread to its neighbours.
- After each round of iteration, both probabilities decay with factor λ_{true} (resp.λ_{false}). - We iterate this algorithm until the end of the propagation.

We implemented this algorithm with a Breath First Search structure to simulate news time-propagation. Regarding the propagation probability from one node at depth d to one of its neighbor node, the detailed formula for fake information is as follow:

$$P_{propagation} = p_{false}(1 - \lambda_{false})^d \tag{1}$$

For true news, you we have the same formula, with p_{false} to p_{true}, and λ_{false} to λ_{true}.

Nb : Thanks to the decaying factor, we are certain that the propagation is going to stop at some point. This phenomenon appears in real life thanks to the platforms' algorithm, which is always biased towards to newer content.

Finally, we build the cascade (in the form of a prefix tree) that represents the news spread in the network.

Please see below a news spreading in a 30 nodes Barabási–Albert network (and the resulting cascade) :

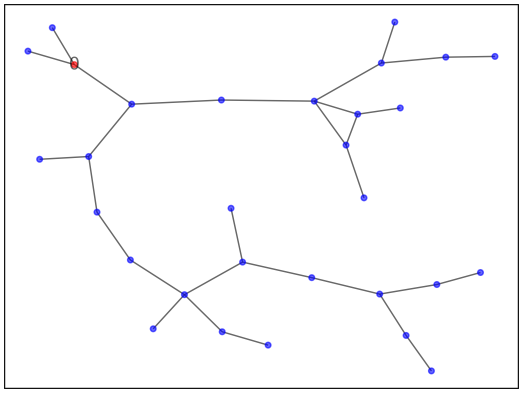

Figure 6: Initialization step

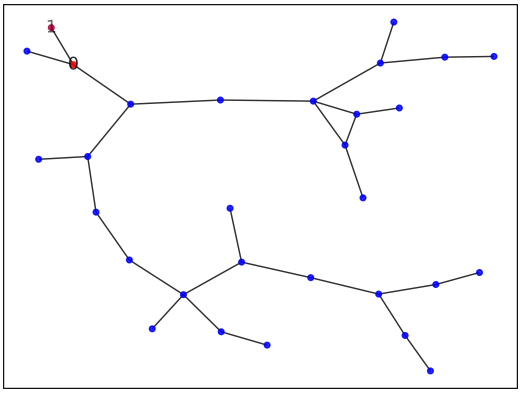

Figure 7: t=1

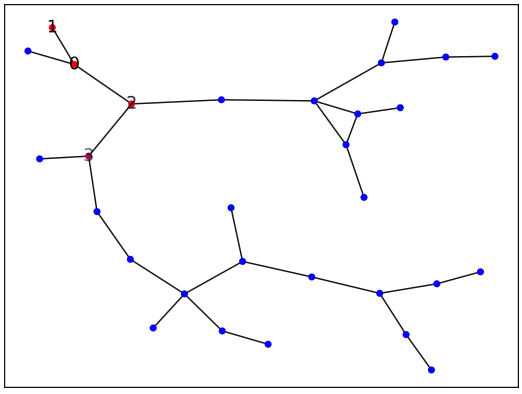

Figure 8: t=3

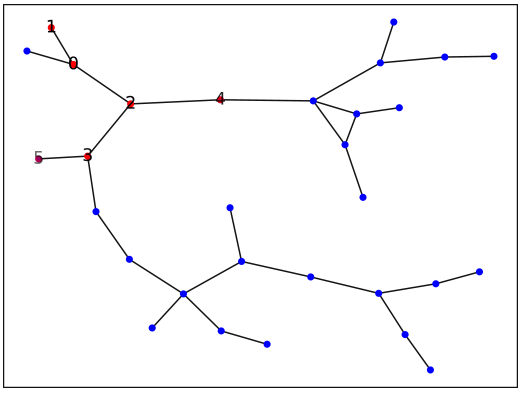

Figure 9: t=5

And for this propagation instance, we get the following cascade :

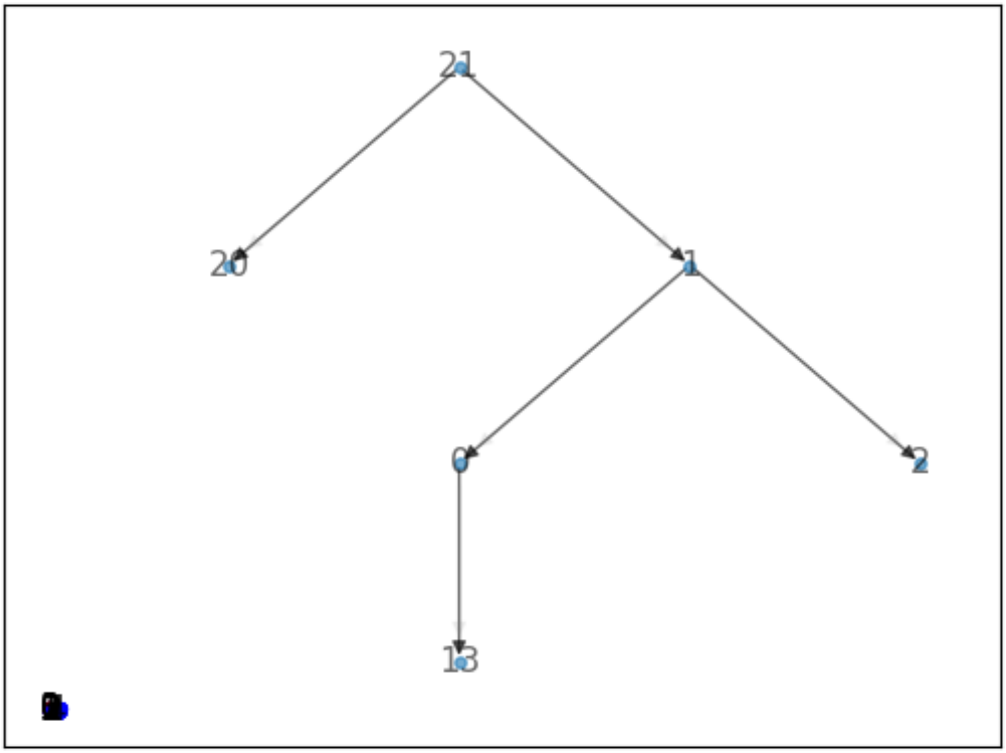

Figure 10: Cascade of a news propagation

This graph looks quite simple, but it has only an illustrative purpose. Now that we have simulated our model, we can grow it arbitrarily large, and make any further hypothesis.

4.3 Propagation simulations in a Watts-Strogatz network

We decided to base our study on a Watts-Strogatz network of 500 nodes (or people) with each node having in average 20 friends. This represented a good balance between a complex enough network and computation time.

The graph below illustrate both the clustering and small world aspect, with regards to the rewiring probability p.

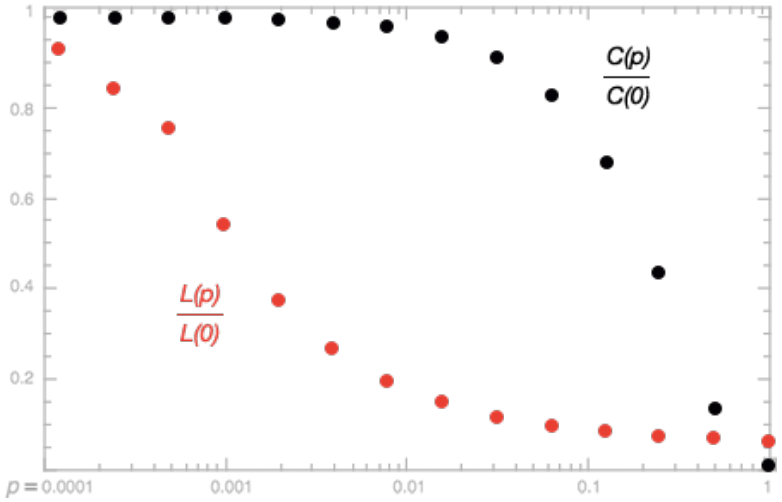

Figure 11: Influence of p coefficient on W-S graph properties

Where L(p) is the average path length, while C(p) is the local clustering coefficient.

For our network model, we have chosen a rewiring probability of $p = 0.01$. This value allows for both properties to co-exist in the graph, which is the closest to what happens in real life.

Once the network generated, to create our dataset we simulated 1000 fake news propagation and 1000 true news propagation.

Regarding the choice of the parameters p_{false}, p_{true}, λ_{false} and λ_{true}, we processed as follow. We generated p_{false} and p_{true}, using a gaussian normal function centered on respectively 0.5 and 0.3 with a variance of respectively 0.2 and 0.1. Indeed, fake news have a higher probability to propagate from one node to another one. We have chosen this model to replicate the fact that some news are more popular and thus more likely to spread than others. For the decay factor, we chose the value 0.4 to limit the depth of propagation graphs. We found out that those values enable us to obtain similar results as Twitter propagation graphs as in real life, news don't spread as much depth-wise, as new content is generated continuously.

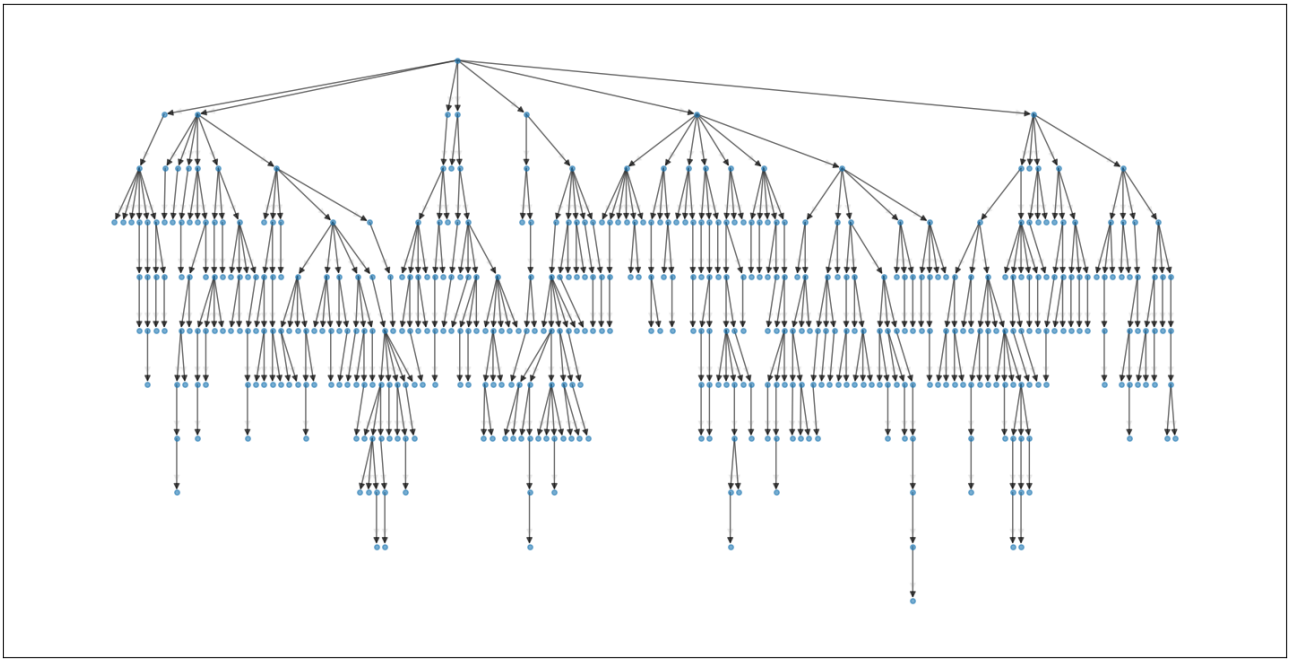

Figure 12: Propagation of a fake news in a Watts-Strogatz network of 500 nodes

4.4 Propagation simulations in a Facebook network

Now that we have a working model, let's try to apply it on real world data. To that effect we are going to use a dataset from SNAP (Stanford Network Analysis Project) of ego circles on Facebook [16]. This dataset represents 4039 nodes and 88234 edges, each node representing a person and an edge a friendship connection.

Even though this dataset is only a (very) small subgraph of Facebook, it possesses some interesting properties. Indeed it is connex (meaning there exists a path between every pair of nodes). It has a diameter (longest shortest path) of 8. this might seem quite long for a social network, but the 90-percentile effective diameter is only 4.7, which illustrate well the small world phenomena.

Moreover, its degree distribution follows a power law (see below)

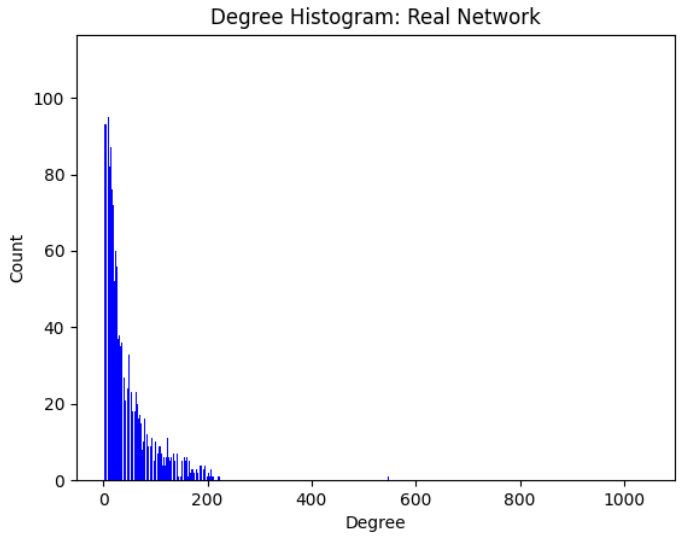

Figure 13: Degree Distribution of the Facebook Network

The clustering coefficient of this graph is 0.65, which is not so high. That is the only limitation we have regarding what we were expecting studying a real life network.

Now, let's run our news propagation algorithm on this network, for both fake and true news. We get the following cascade :

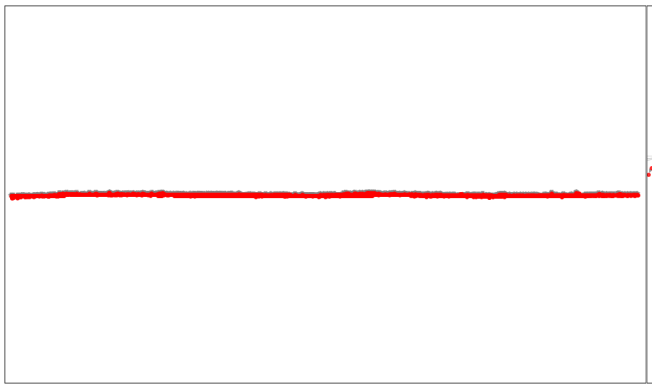

Figure 14: Full Scale Cascade

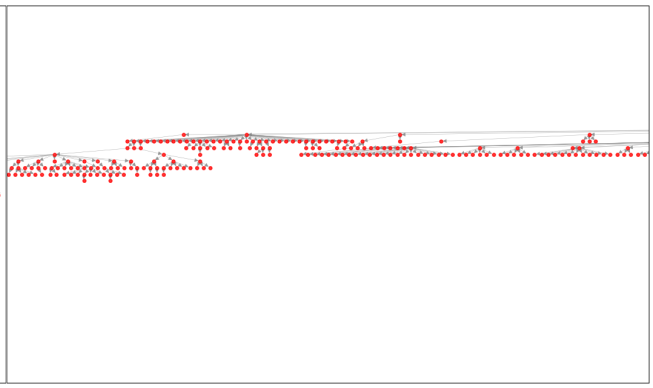

Figure 15: Zoomed-In Cascade

We notice that we get a very wide cascade. It echoes what might happen in real life when a topic makes a buzz and reach a lot of different people. The connected structure of such social networks allows for high virality.

In the next chapter, we are going to observe how changes in starting hypothesis impact the news propagation and classify the news accordingly.

5 Fake news / True news classification

Now that we have all the different cascades for both true and fake news, we need to identify key features. We are going to compute every features for each propagation cascade. The cascades are stored as NetworkX DiGraph objects (directed graphs). This library makes computing features relatively easy. We are going to compute the features for each graph and store them in a pandas dataframe (the dataset) along with their label (True news or Fake News). Finally, we will feed this dataset to a machine learning model to classify news with regards to their veracity.

5.1 Selecting features

We selected our features based on the article [14] and added other basic features. There are two main features categories : Structural and Temporal. We have chosen in a first approach to focus more on structural features as they are easier to compute and we believe their impact is going to be greater. Among structural features, we have chosen : number of nodes, depth of the cascade, maximum breadth, structural virality measure as introduced by [15], and average number of neighbours. The structural virality is a measure that differentiate content spread through a single, large broadcast, and content spread through multiple generations with multiple individual responsible for the diffusion. It is computed with the following formula : $\frac{1}{n(n-1)} \sum_{(i,j) \in G} d_{ij}$, where d_{ij} is the minimal distance between nodes i and j

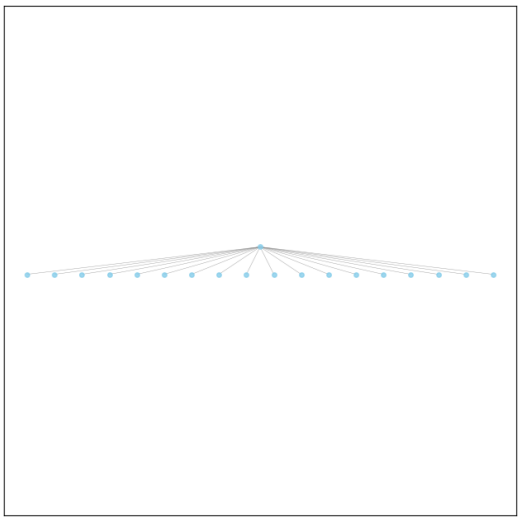

Figure 16: Broadcast Virality : 1.89

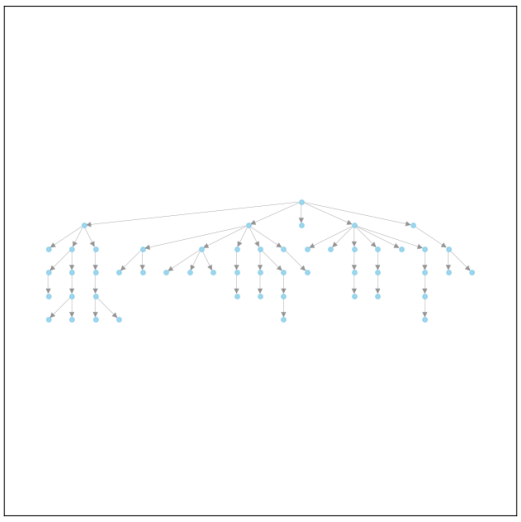

Figure 17: News Virality : 5.06

We implemented algorithms to retrieve the value of such features from each propagation graph from the simulated propagation dataset. We obtained a database that can be fed to a machine learning model.

5.2 Choosing the model

As for the classification model to be chosen to distinguish true news from fake news, we considered using different kind of models: logistic regression, SVM with linear or non linear kernel, or neural network.

Support Vector Machines (SVM) seems to be a great fit for this kind of classification where the number of feature is quite small, and the number of training data intermediate ($\sim$ 2000 training data). Thus, we decided to implement SVM with linear kernel as well as SVM with polynomial kernel, and with Radial Basis Function kernel to compare the results of these models. It is worth mentioning that the most popular RBF kernel is the wildly used Gaussian kernel.

In linear SVM, a hyperplane is selected to separate the points in the input variable space by their class, with the largest margin. As real data cannot be perfectly separated, we also define a variable C which quantifies the amount of violation of the margin allowed (regularization parameter). The lower C, the more sensitive SVM is to training data. For SVM with kernel, the idea is to apply a kernel function (e.g Gaussian Kernel) to the input data before running the SVM. We used scikit-learn library which offer SVM functions to implement our models.

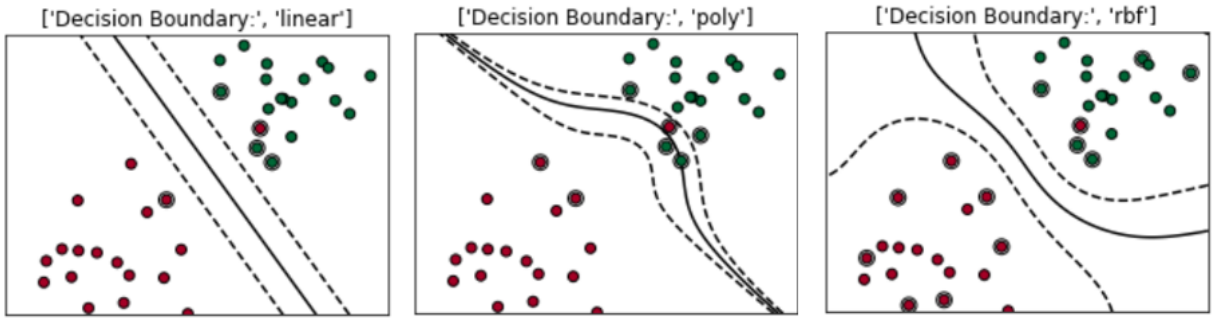

Figure 18: Decision boundaries for linear, polynomial, and radial basis function

5.3 Classification results in a Watts-Strogatz network

After training on different SVM models, we obtained the results bellow. For the metrics, we chose the accuracy, the precision, and the recall as they are the most used metrics. Nb : The precision is the number of true positives divided by all the number of predicted positives, while the recall is the number of true positives divided by the number of actual positives.

Linear Kernel:

- Accuracy: 0.7225
- Precision: 0.6755725190839694
- Recall: 0.8719211822660099

Radial Basis Function:

- Accuracy: 0.6975
- Precision: 0.6506550218340611
- Recall: 0.7842105263157895

Polynomial of degree 2:

- Accuracy: 0.755
- Precision: 0.6928838951310862
- Recall: 0.9203980099502488

Polynomial of degree 3:

- Accuracy: 0.6725
- Precision: 0.6386554621848739
- Recall: 0.7715736040609137

From this test we infer that using an SVM with a Polynomial of degree 2 enables us to obtain the best results. Indeed, the accuracy, the precision, and the recall are higher for this model than for the others. The Linear SVM also seems to be a good fit for our dataset and we manage to get an accuracy of 72.2% as compared to the 75.5% in accuracy obtained with the Polynomial of degree 2.

5.4 Classification results in a Facebook network

Finally, we ran our algorithms on the Facebook network. Before looking at the SVM accuracy, let's look at the values of some features:

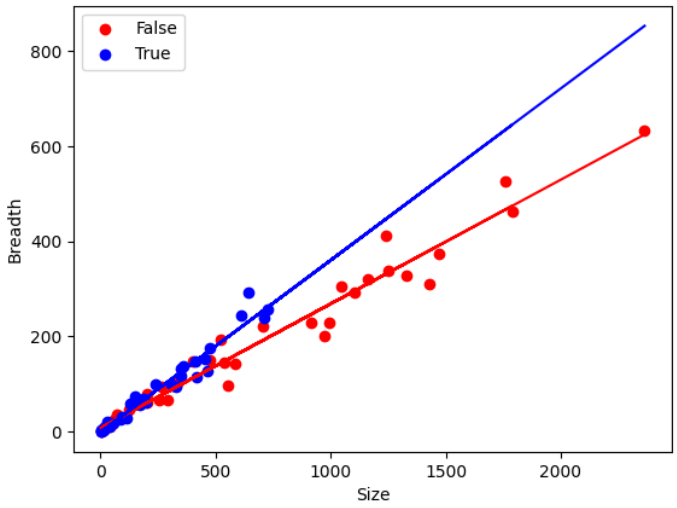

Figure 19: Size vs Breadth for both True and Fake news

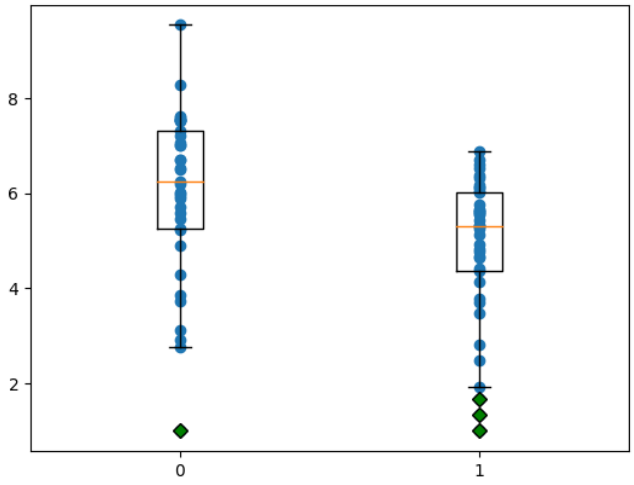

Figure 20: Structural virality for both True (1) and Fake (0) news

We indeed see structural differences between the two propagation scheme.

Now, let's use the SVM to try and classify the data, leveraging the structural differences in propagation for the Facebook dataset. We ran the SVM with Polynomial of degree 2 and we obtained the following results:

Facebook Result:

- Accuracy: 0.69
- Precision: 0.7
- Recall: 0.78

The accuracy of our model is quite good. We manage to classify news with an accuracy of 69%.

6 Project Takeaways

In this project, we have demonstrated a simulated proof of concept for our approach. When there are real differences of propagation between True and Fake news, collecting cascades and using supervised learning approaches such as a SVM allows the classification of such news with a good accuracy.

This approach could prove to be useful in overseeing fake news diffusion on social media. It would nevertheless require an active overseeing of trending news, in order to stop the diffusion of fake news as soon as possible.

6.1 Technical Analysis

During this project, we learned how to scrap a social media using Python libraries and the social media API. This project also enabled us to acquire an in-depth understanding of network theory, and classification techniques. We manipulated graphs using NetworkX library, dealt with data using pickle and Pandas libraries, and implemented SVM models using Scikit-learn. Over all, we learned how to manage a machine learning project: we harvested data, selected and computed features, created a database, implemented classification models, and obtained satisfying classification results.

6.2 Difficulties encountered

In this project, we have encountered many difficulties that have slowed down our progress. We have been able to solve some, but not all of them :

6.2.1 Technical difficulties

- Lack of existing Database

The first difficulty that we have encountered in our project was the absence of an existing database. Indeed, we have chosen a very peculiar approach that was not explored much before. Most of the existing Fake News Detection algorithm rely on Natural Language Processing (NLP). Hence, most of the databases focus more on Tweets' actual content and not so much on their propagation.

Facing this issue, we have decided to build our own database. This was the way to move forward, but it came with its own struggle.

- Twitter API limitation

Our first idea to build the database was to rely on the Twitter API to scrape tweets and build the cascades (1 cascade being a data point). We faced two limitations.

The first one is the allowed rate : Twitter has installed a bottleneck on the number of tweets one can scrape per hour. This would have prevented us to collect the number of tweets we would have needed in a reasonable time.

The second, bigger, issue was the fact that Twitter doesn't keep track of every retweets. It only keeps available for scrapping the latest 100 retweets of every tweet. This would have prevented us from getting the beginning of the cascade, when the news is still "hot", the most interesting part for our analysis.

We managed to by-pass this issue by using a Python Library called "twint", it allows for scrapping without rate limitation. The only issue is that it doesn't keep an exact tracking of

retweets. We then developed a matching algorithm to reconstruct the cascade.

- Python to CSV integer storage

Initially we stored our data in CSV files : it made it easier to consult the data and make initial analysis. But we noticed that the primary key we were using for our database (the tweets' index) were not stored properly in those formats. Indeed, sometimes the last digits of the id were erased, or rounded up. It made exact search impossible, and thus broke our algorithm.

To solve this issue, we went with another Python Library called "Pickle". It allows us to store "Pandas" database in pickle files, with exact (or at least sufficient) integer precision. It seems like a good fix for now as we haven't had any issues with it for now.

But the issues we faced on this project weren't only technical, we also faced some project management difficulties as well.

6.2.2 Project Management difficulties

- Work from home setting

Obviously the current situation has been a major hurdle for our project advancement. It has not been possible to meet "in-person" between ourselves and with our referent teacher.

Nevertheless, we managed to solve it by scheduling recurrent Video-Chat calls. Even thought they are not as effective to communicate and think about new ideas, they allowed for continued progress throughout this project.

- GitHub Version Planing

In order to both work at the same time on project files, we used the Version Controller Git. We were having some issues making it work and keeping track of the changes at the beginning, given that we were not familiar with Git.

But with time and the help of Google, we have gotten a hand on it and it ended up working much better. It has been quite practical to track every changes that was being made and not work on different versions of the same file.

- Time Management (Internship interviews, Course-related work)

The Third Year of the cursus at CentraleSupélec is very intensive. The work load is quite high and at times, we have been struggling to keep up, especially before and during exam weeks. On top of that, the search for an end of study internship has proven to be quite challenging, especially with COVID. Hence, it was sometimes difficult to find time to make decent progress.

In order to fix this issue, we started to make more detailed to-do lists and overall planning. It allowed us to keep track of our progress and keep us motivated in this challenging time.

6.3 Eventual Next Steps

While satisfying, the end result of this project is not finished, given all the difficulties encountered. Were we to have more time, there are several areas that would require deeper analysis. In particular, there are three main aspects we can work on to better its accuracy : dataset quality, features selection and model design.

a) Quality of the dataset

This would be the most important part, but also the most difficult. Machine Learning approaches require a lot of data in order to be trained properly. The rate bottleneck on the Twitter API limited our ability to build a proper dataset. With more time, one could build a more comprehensive dataset of both actual True and Fake News to plug into the SVM for a proof of concept.

b) Model Features

Right now, we have selected basic features to be extracted from each news. The selection process was mainly based on the article [13] which describes some structural propagation differences between real and fake news for some given features. One way to pursue our study would be to fine tune such features (by trying to decorrelate them), exploring new features (maybe try to look at a temporal aspect as well) and ultimately choosing the right features for the model.

c) Model Design

Currently, we have chosen to use a Support Vector Machine (SVM). This supervised learning algorithm allows for online training and decision making. But maybe this model is not the most efficient or the best adapted to our situation. If we had more time, we might have explored other classification method. In particular, we could have looked at a unsupervised learning using deep learning.

7 Conclusion

To conclude, while there still are lots of challenges ahead, we've built a simulated basis to support our study and confirm the theoretical backing of [13]. Which is definitely a good result. Possibly, the difficulties we encounter in building the database may be the reason why our approach wasn't used already. Nevertheless, building such model might prove to be relevant in the fight against disinformation on platforms. Beside, this approach is currently under scrutiny by official agencies, as shown by [14] : the CSA (Conseil Supérieur de l'Audiovisuel) has just released a report on "Fake News Virality on Twitter".

8 Bibliography

[1] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang and Huan Liu. Fake News Detection : A data mining perspective.

[2] Natali Ruchansky, Sungyong Seo, Yan Liu. CSI: A Hybrid Deep Model for Fake News Detection

[3] Carlos Castillo, Mohammed El-Haddad, Jurgen Pfeffer, Matt Stempeck. Characterizing the Life Cycle of Online News Stories Using Social Media Reactions

[4] Chengcheng Shao, Giovanni Luca Ciampaglia, Alessandro Flammini, Filippo Menczer. Hoaxy: A Platform for Tracking Online Misinformation

[5] Zhiwei Jin, Juan Cao, Yongdong Zhang, Jiebo Luo. News Verification by Exploiting Conflicting Social Viewpoints in Microblogs

[6] Zhiwei Jin, Juan Cao, Yu-Gang Jiang, Yongdong Zhang. News Credibility Evaluation on Microblog with a Hierarchical Propagation Model

[7] Simon Lorent. Fake News Detection Using Machine Learning

[8] Hunt Allcott, Matthew Gentzkow. Social Media and Fake News in the 2016 Election

[9] Lilian Weng, Filippo Menczer Yong-Yeol Ahn. Virality Prediction and Community Structure in Social Networks

[10] Devesh Varshney, Sandeep Kumar, Vineet Gupta. Predicting information diffusion probabilities in social networks: A Bayesian networks based approach

[11] Adrien Guille, Hakim Hacid, Cécile Favre, Djamel Abdelkader Zighed. Information Diffusion in Online Social Networks: A Survey.

[12] Soroush Vosoughi, Deb Roy, Sinan Aral. The spread of true and false news online.

[13] Soroush Vosoughi, Mostafa 'Neo' Mohsenvand, Deb Roy. Rumor Gauge: Predicting the Veracity of Rumors on Twitter.

[14] CSA, La Propagation des fausses informations sur les réseaux sociaux: étude du service Twitter.

[15] GOEL, Sharad, ANDERSON, Ashton, HOFMAN, Jake, et al. The structural virality of online diffusion. Management Science, 2016, vol. 62, no 1, p. 180-196.

[16] J. McAuley and J. Leskovec. Learning to Discover Social Circles in Ego Networks. NIPS, 2012