Fake News detection using Machine Learning on Graphs - Final Report

Arnaud Autef
Department of Management
Science and Engineering
arnaud15@stanford.edu

Alexandre Matton
Institute of Computational
and Mathematical Engineering
alexmt@stanford.edu

Manon Romain
Institute of Computational
and Mathematical Engineering
manonrmn@stanford.edu

Abstract

Fake News is an emerging topic that has received a lot of attention since the 2016 US presidential election, where many reckon that the spread of false information on social networks had a significant influence on its outcome. Most of the current work focuses either on the actual content of the news articles or the user that shares the news article on social media. However, social media platforms where fake news spread can be easily modeled as graphs and the goal of our project is to leverage techniques from *Machine Learning on Graphs* for design better models for fake news detection. Using a range of techniques (the SEIZ contagion model, Graph Neural Networks, pre-trained language models for text features extraction...), we obtained results close to or outperforming the state-of-the-art on the Fake News detection datasets Twitter15/16, illustrating the benefits of graphical modelling. However, our models are prone to overfitting on those small datasets, and there is room for further improvements. Finally, our work highlights the need for better datasets for Fake News detection on social media.[1]

1 Introduction

We propose to tackle the problem of detecting fake news using the graph representation of its diffusion through social media.

As the challenge of fake news detection is gaining leverage, the research community is starting to gather data from various sources to try to define the bounds and goals of this wide research. Some well-structured datasets have emerged and baselines are getting more and more compared across the most common ones.

In our case, we chose to tackle supervised graph classification. Our graphs are extracted from Twitter. A piece of news, spreading verified or false information, is shared in a root tweet. This first action triggers retweets and comments. All those actions have a undeniable graph-like structure. Therefore given a graph and its metadata (see Section 3 for more detailed explanation), we would like to quantify to what extent it is possible to predict the "trustworthiness" of the initial news.

[1] Our code is publicly available on Github: github.com/manon643/FakeNews

Preprint. Under review.

2 Related Work

This problem is well-known and several papers have already studied this problem. However, they vary in their choice of data modeling.

Ma et al. [9] introduced some important datasets and offered a first solution, using text content only. They framed it a sequential problem by transforming each example to a sequence of chronologically ordered tweets. They transformed each text to a fixed-size vector using word embeddings for the k-most important words of each tweet according to TF-IDF. On top of that, they used a multi-layer RNN on the embedding sequences to compute the label.

A lot of other solutions use this sequential format as it has been proved efficient. Liu and Wu [4] use a combination of RNNs and CNNs on user features instead of text to achieve better results.

Some methods using more advanced models have recently emerged. For instance, Ma, Gao, and Wong [10] use Tree Recursive Neural Networks and improve on their past results with the exact same data. The same team also experimented with recent deep learning techniques, such as Generative Adversarial Networks [7].

Except for the Tree Recursive Neural Networks, all these solutions do not really make use of the graph structure induced by the tweet history. Yet, there might be a lot of useful information hidden in the structure of these graphs. We observed that propagation graphs' shape depends on their label. Fake news are often striking and people are more inclined to react to them, so the speed and number of tweet/retweets are not the same too. Our aim is to try to take advantage of this structure to use it in our models.

3 Data overview

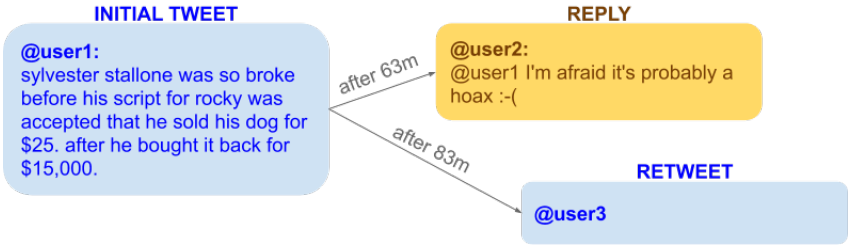

Figure 1: Example of **fake** news sharing tweet and subsequent retweets on Twitter16

The datasets we work with are *Twitter15* and *Twitter16* [6, 9]. These two datasets share the same exact structure. Both of them contain the tweets and retweets from a thousand of news articles published in 2015 and 2016. For each news article, the data contains the first tweet that shared it on Twitter, and a sequence of retweets following this initial post. We show one such data point (initial tweet and first two retweets) on Figure 1. Each event is labeled according to the initial news article, the label is taken out of four possible classes: "true", "false", "unverified", "non-rumor". Labels are evenly distributed in both datasets.

The task we address is *supervised graph classification*: our models label each event between 4 possible classes using the tweet/retweet data at hand. We highlight below the fundamental graphical structure of this data, that our models strive to leverage.

For a given news article, the sequence of tweets and retweets subsequently observed on Twitter is a tree:

1. Nodes correspond to pairs of Tweet and Twitter User IDs. A retweet without comment shares ID of the initial Tweet, we use the User ID to distinguish them.
2. We put an edge between node 1 and 2 if node 2 is a retweet (with or without comment) of node 1.

Data cleaning Twitter15/16 were not perfectly collected and we spotted some data quality issues (e.g. negative propagation time, lines that were present twice). We implemented some fixes to get an

acceptable level of data quality. For instance, we made sure that all propagation times are positive and that the tree structure always holds.

The dataset already contains the text written with each tweet or retweet. Due to privacy issues however, features of the user behind a tweet are not directly available in the Twitter datasets. We augmented our dataset by retrieving them using the Twitter API via the `tweepy` library. A major drawback to this is that we could only get the current state of the users (*i.e* as of October 2019). Hence, we had to make the strong assumption that user features did not change too much since 2015, or change in a way that doesn't affect too much our downstream classification task. Twitter15/16 are standard datasets for Fake News detection, and all papers published since 2016 must have made the same assumption with user features.

As it is usually done in papers using Twitter15/16 for Fake News detection, we hold out 10% of the events in each dataset for model tuning (validation set), and the rest of the data is split with a ratio of 3:1 for the train and test set.

Text features To get the best of the text data, we cleaned it before feeding it to a Transformer-based model. We tried BERT [1] and RoBERTa [5], and went with the latter option as it is the one which provided the best accuracy. Transformer-based models take as input a sequence of words, tokenize it, and output a list of highly-dimensional token embeddings. Right after tokenization, some starting and end tokens are added at the beginning and end of each sequence. We extract the final embedding of the starting token to get a fixed-dimension vector out of each sentence, as it is done in [1]. This 768 dimensions vector supposedly gives a representation of the sentence meaning. This is currently the state-of-the-art in word sequences embedding.

User features Some users deleted their accounts between the creation of Twitter15/16 datasets and when we started the project. Hopefully, we still managed to get the data for 90% of the users. We filled in the blanks with sensible values: medians for some numerical features, 0 for others, etc. Aggregated features used to fill out the blanks (e.g. medians) were computed from users of the train set solely to avoid data leakage. The features we extracted are presented on Table 1.

Name	Type	Description
created_at	numeric	Normalized time since account creation.
favourites_count	numeric	User favourites count.
followers_count	numeric	User followers count.
friends_count	numeric	User friends count.
geo_enabled	boolean	User geographical location enabled or not.
has_description	boolean	User has description on Twitter profile or not.
len_name	numeric	Length of the User username on Twitter.
len_screen_name	numeric	Length of the User name on Twitter, as seen by other users.
statuses_count	numeric	User count of statuses.
verified	boolean	Verified Tweeter user or not.

Table 1: Outline of our User features for Fake News classification

Trees extracted from Twitter15/16 The trees we defined (see above) from our news articles and Twitter data do not show a complex structure and are mostly "flat" (with low depth). On Twitter15/16, the maximum depth of the root tweet / retweets trees we build is around 5, and most nodes are simply at depth 1 or 2 from the root node. We illustrate this problem on Figure 2.

News article: a Twitter tree or a forest? We must mention an important limitation of the Twitter15/16 datasets we realized. For a given (Fake / True) news article, a *single* "root tweet" sharing this news article on Twitter is retrieved in the data, and we end up with a single tree of retweets. In reality, we expect that any important news article is shared independently by several users on Twitter ("independently" meaning here that they do not retweet one another). In Twitter15/16 we miss all those alternative "root tweets" as only one is selected, and lose a lot of graphical information: we have a single tree instead of a "forest".

The Twitter data collected by Ma et al. [9] to build Twitter15/16 is thus partial. This crucial limitation has to be kept in mind, but we also acknowledge how difficult data gathering must already have been for Twitter15/16.

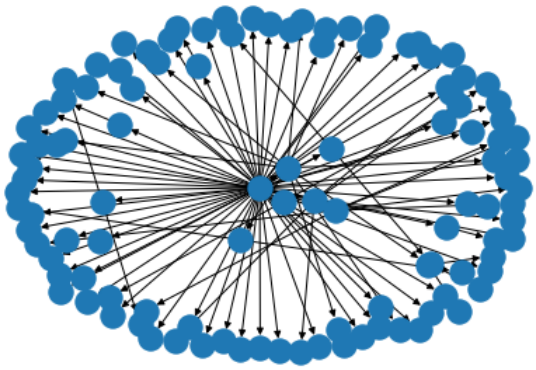

Figure 2: Tree sampled from the Twitter15 dataset and visualized on Networkx

4 Baseline Models

Here, we describe briefly the baseline models we used to compare to our models leveraging techniques from Machine Learning on Graphs.

4.1 Gradient Boosting with Decision Trees (GBDT)

We first built a baseline using the *user features* of tweets, in the spirit of Liu and Wu [4]. Indeed, their results are "state-of-the-art" for Fake News classification on Twitter15/16 and we would like to see if we can get similar results with our models. However, we note that their code is not publicly available and hard to reproduce.

We opted for a GBDT model for two main reasons: user features are "tabular" in nature and boosted tree models perform well on those; GBDT is quick to implement and tune for good results. We use the Python API of the efficient LightGBM implementation of GBDTs [11].

To train the GBDT model, data is processed very simply: for each news article's series of retweets, we aggregate user features from Table 1 using user IDs for each retweet (mean aggregation for numerical features, sum aggregation for boolean features encoded as 0 or 1), the time (in minutes since the root tweet initial post) at which each retweet occurs is also mean-aggregated and used as a feature.

We improved on the baseline GBDT model by adding additional features from a fitted graphical contagion model (SEIZ: see Section 5.1) for each tweet/retweets propagation observed on Twitter.

4.2 Long Short Term Memory (LSTM) Network and Multi-Layer Perceptron (MLP)

In their paper, Liu and Wu [3] obtain best results with a combination of Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) to classify news articles. The idea being that their RNNs and CNNs capture the "sequential" aspect of the data at hand: from an initial tweet sharing a news article, retweets and reactions from Twitter users are observed sequentially as time elapses and the information propagates on Twitter. Intuitively, their RNN model captures "global" information from this temporal tweets propagation while their CNN model captures "local" information.

In [3], authors employ GRU (Gated Recurrent Unit) cells for their RNNs. We opt for LSTM cells, a more common RNN cell, with arguably better representation power but higher complexity. Following their work, the sequence fed to our baseline LSTM corresponds to the sequence of retweets that follows the root tweet for any of our data point, earliest retweets first. Doing so, the sequence obtained for each data point is of *variable* length. We deal with this characteristic as Liu and Wu: if this length

is below a threshold $L = 40$, authors randomly oversample some user features in this sequence to reach threshold size L; if the sequence longer than L, it is truncated to size L.

For the project milestone, we represented each retweet in the sequence via the user feature of the user retweeting. Those features are tabular and hard-to-learn-from for a LSTM (we had to properly scale them to obtain reasonable results). For this final report, we experimented with the Transformer-based features of the retweets text, those *dense* 768-dimensional embeddings are easier to learn from for a LSTM and we obtained better result.

We also train a MLP on the text feature of root tweets. By comparing this bare baseline to the LSTM, we could evaluate nicely the benefits from capturing the "sequential" nature of the retweets data with a LSTM.

The baseline LSTM model is then compared to Graph Neural Network (GNN) models to evaluate the additional benefits from capturing the "sequential" *and* "graphical" nature of our retweets data (trees in our modelling, as exposed above).

5 Graphical Models

5.1 SEIZ Contagion Model for Fake News Detection

The SEIZ model is probabilistic contagion model that has been applied to Tweets by Jin et al. [2]. SEIZ initials stand for the different states in which Twitter user can be over time, with respect to a news article propagating on the social network:

- S: Susceptible. In theory, all active Twitter users that can reasonably be in contact with tweets associated with the news article.
- E: Exposed. Twitter users exposed to tweets associated with the news article.
- I: Infected. Twitter users believing the content of the news article.
- Z: Skeptics. Twitter users that do not believe the content of the news article.

A graphical model defines the possible transitions between those intuitive states. The attributed of this graph and the initial populations in each of the SEIZ states govern the dynamics of their populations over time, according to an Ordinary Differential Equation (ODE). The graphical model is reproduced on Figure 3 as seen in class.

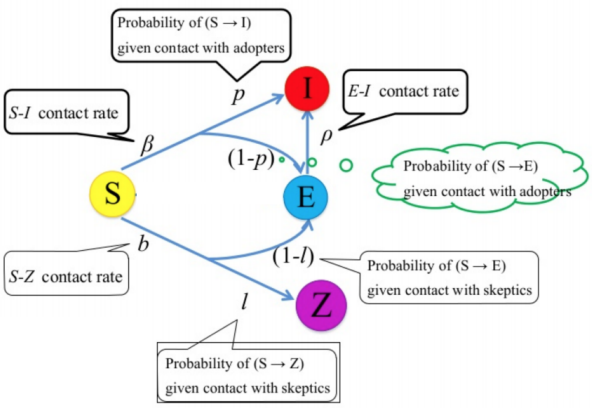

Figure 3: SEIZ graphical model

In our work, we followed the approach of Jin et al. [2] on our specific data. For each news article in Twitter15/16, the input to Jin et al.'s procedure is the number of distinct retweets (distinct `(user_ID, tweet_ID)` tuples) observed over time. We remind their procedure (for a single data point i.e. number of retweets over time) below:

1. Start from random SEIZ parameters and initial populations

2. Find the best fit (in the least squares sense) of the obtained I(t) curve to our number of retweets over time. We use the *least_square* function from Scipy optimize for optimization, and implement ourselves a simple Euler method to solve the ODE (following the approach of the authors).

Examples of fit obtained on Twitter15/16 are presented on Figure 4. We restrict the fit of the SEIZ parameters to the first 120 minutes after the inital root tweet is posted on Twitter. It took us several hours to compute the "best" SEIZ parameters and initial populations for the roughly 2000 news articles of Twitter15/16.

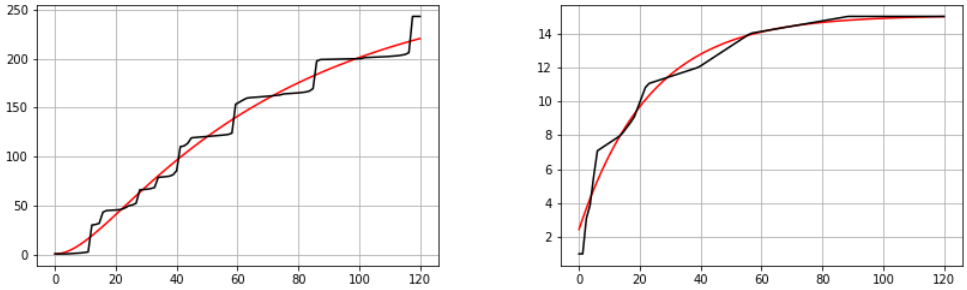

Figure 4: Example SEIZ fit on Twitter15 (left) and Twitter16 (right). The red curve is the SEIZ fit for I(t) and the black curve is the number of retweets observed over time.

The "best" parameters obtained (10 in total) for each news article are used for classification. We employ a GBDT model on this structured tabular data for best results. In the results section, we compare this GBDT SEIZ approach to the baseline GBDT approach using user features only. We also evaluate an *ensemble* of the two models to see if it can compete with Graph Neural Networks (GNNs).

5.2 Graph Neural Networks

Finally, we experimented with GNNs, our best performing models. GNNs are powerful neural network models to obtain node embeddings in a graph. In our 4-class classification problem, the graphs are the trees described in Section 3 and the task becomes *graph classification*.

We experimented with 3 variants of Graph Neural networks: Graph Convolutional Networks (GCNs), Graph Attention Networks (GATs) and GraphSAGE. We evaluated our model every epoch and selected the model with best validation accuracy.

As node features to any of our GNN models, we concatenate the User Features and (BERT-based) text features detailed in Section 3. For a given news article to classsify, we obtain GNN-based embeddings for all nodes of its retweets tree. For graph classification, we then need to aggregate those node embeddings: common approaches include max-pooling, mean, or sum aggregation. Because each tree has a root node that is directly related to the news we want to classify (see Figure 5), we thought that using the output embedding of this node would be useful. However, if the number of GNN layers is small, this embedding doesn't take into account nodes that are deep in the retweets tree. Thus, we decided to concatenate this root node embedding with the mean and max aggregates of all nodes of the tree, so that we could get a more accurate representation of the tree. We obtained better results with this approach.

6 Experiments and results

6.1 Experimental details

We ran two sets of experiments: a first set of experiments on the multi-class problem ("true", "fake", "unverified", "non-rumor") and another set of experiments on the binary classification problem (retaining only news labelled "true" or "fake"). Reference papers usually carry out those two types of

experiments.

GNN experiments We compared the three models (GAT, GraphSage, GCN) and did an ablation study where we examined the respective benefits of user and text features. We have three different sets of features in our ablation study: `text_only` using only the BERT-based text features, `user_only` using only User features described in Table 1, and `all` that uses both. Results are presented on Figures 5 and 6.

For the experiments, we used 1 to 2 layers, a batch size of 32, a dropout probability ranging from 0.1 to 0.7 and an AdamW optimizer. Specific hyperparameters were tuned for each model and dataset based on the *validation* set classification accuracy.

Gradient Boosting experiments We trained three models: a baseline GBDT model leveraging (engineered) User features, a GBDT model leveraging SEIZ features, and an ensemble model of the two.

We used 2000 tree learners, a maximum depth of 5 for the decision trees grown, and a learning rate varying between datasets (Twitter15/16) and models. The best learning rate was selected case by case to optimize accuracy on the *validation* set.

6.2 Competitive results on multi-class classification

	Twitter15			**Twitter16**		
Split	**Train**	**Val**	**Test**	**Train**	**Val**	**Test**
Recursive Tree[8]	NA	NA	0.723	NA	NA	0.737
RNN+CNN[3]*	NA	NA	0.842	NA	NA	0.863
GBDT_user	0.962	0.629	0.628	1.00	0.671	0.647
GBDT_seiz	0.672	0.412	0.360	0.741	0.506	0.377
Ens_GBDT	0.959	0.635	0.577	0.995	0.617	0.618
MLP text	0.931	0.568	0.536	0.882	0.634	0.549
LSTM text	0.899	0.584	0.622	0.922	0.622	0.587
GraphSage text	0.954	0.624	0.622	0.866	0.756	0.712
GCN all (Our best)	1.00	0.719	0.690	0.859	0.841	0.750

Table 2: Final results: Accuracy of baselines and graphical models on 4 classes classification. Results from top reference papers are featured for comparison. *: no code available and not reproducible

Results are presented in Table 2. We divide models in (1) baselines from the litterature [3, 8], (2) Gradient Boosted models, (3) Neural models (4) our best GNN mode: a GCN using both user and text node features. We highlight that, despite their high scores, Liu and Wu [3] did not make their code available.

Overall performance on Twitter15 On this dataset, our performances are competitive with state of the art (excluding [3]). We can notice heavy overfitting on all variants of our model (see Figure 5) that could maybe be addressed with dropout (we didn't have time to exhaustively grid search the best parameters).

Overall performance on Twitter16 Here, our model outperforms the current state of the art (excluding [3]). Note that the size of this dataset is approximately half of the first one, so results (for every model) are subject to higher variance. It also induced higher overfitting on the validation set as can be seen on Figure 6, a trend we didn't notice on the first dataset.

Good GBDT performance but limited improvements with SEIZ We note that Gradient Boosted models attain a reasonable level of performances despite their simplicity and ease of training. A GBDT model trained solely on SEIZ features is however less powerful than one trained on User features, despite some features engineering. We believe that the issue of "tree rather than forest" mentioned in Section 3 is at play here. A SEIZ model relies solely on the number of tweets over time in the wake of a given news article, and we only have the number of retweets over time from a single out of the many independent tweets sharing this news article on Twitter.

Clear Benefits from our Graphical Modeling Our comparison of `MLP_text`, `LSTM_text`, `GraphSage_text` illustrates the importance of a graphical model for Fake News classification. A MLP simply leverages the text of the tweets, a LSTM performs better by accounting for the *sequential nature* of the data at hand, and GraphSage performs best by further accounting for the *graphical nature* of the data (which is fundamentally a tree, with timestamps on its edges).

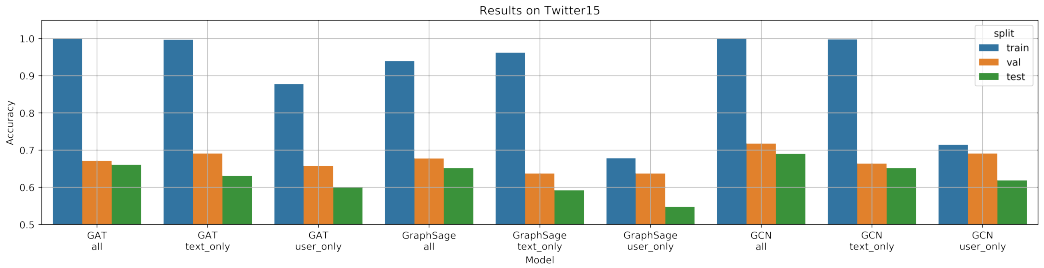

Figure 5: Results on Twitter15

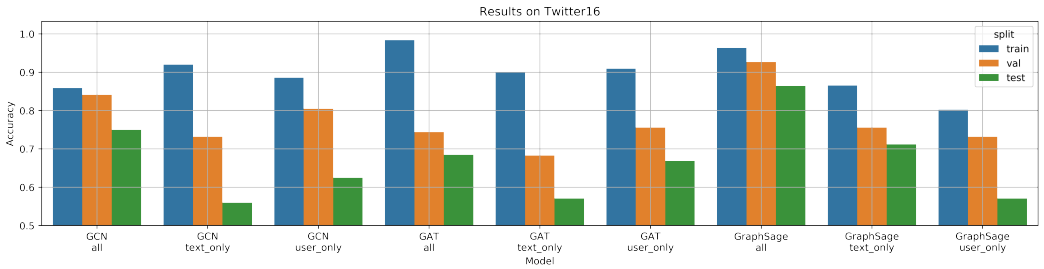

Figure 6: Results on Twitter16

6.3 Top results on Binary classification

This experiment was only run on Twitter15. The current state of the art on the binary classification problem [7] reports an accuracy on the test set of **0.86**. With our best-performing model (here `GCN text_only`), we reach **0.88**. Hyperparameters used for this best model are characterized by a lot of regularization, with only 1 GNN-layer and a dropout of 0.7. Indeed, when shifting from 4 classes to binary classification, we drop about half of the samples on an already small dataset.

6.4 Limits of the dataset and our models

As it can be seen in Figure 5, our graphical models manage to overfit the train set where they get close to 100% accuracy. However, scores on validation and test are significantly lower even if, during training, we never observed a decrease in the validation loss. This implies that our models do not overfit the training set distribution at the expense of the rest of the data distribution, that it the model complex enough to take into account details that are only present in the train set without incurring to much generalization cost. To reduce the gap between accuracy scores on the train set and on the validation and test sets, several options are available.

1. The first and perhaps most efficient option would be to increase the amount of available data. The train set would lose its specificity, and the model complexity would then be used to get a better representation of the data distribution.
2. The second option is to simplify / regularize further the model, so that it gets less affected by the specific details of the training set, and focuses more on important features that generalize to the whole dataset. We tried simplifying the model by reducing the number of hidden dimensions and increasing the dropout rate. This worked out to a limited extent. We are planning on working more on this part.

A shortcoming of the datasets which makes all analyses less reliable is their small size. The datasets contain less than 1,000 trees each, so that the validation and test sets are both very small. From one gradient descent step to another, the accuracy varies a lot on the validation step, making it hard for us to select a model. Even if the learning rate is very small, each tree accounts for close to 1% of the final score. Moreover, the validation and test scores are correlated up to a certain extent, and which makes early stopping pretty useless (especially as the validation loss never goes down). When the model has converged, having a good validation accuracy for a particular epoch is by no means a sign that this epoch is the best for the test score. Finally, from one train/val/test split seed to another, our final scores vary in a range of 5% of accuracy points. It makes precise hyperparameter optimization unfeasible, and we can hardly say whether our results are better or worse than the ones of other papers. We wonder if some impressive results brought forward in [4] may come from a "fortuitous" random seed choice for their train/val/test split.

7 Conclusion and next steps

We came up with a graphical model to represent Fake News data on Twitter15/16. We used this model to train powerful Graph Neural Networks classifiers, leveraging text features (extracted using the recent BERT transformer-based language model) and user features effectively. Their scores are comparable to the state-of-the-art on our datasets, they outperform methods relying on information aggregates, and a simple sequential representation of the tweets data. This work shows that useful information is hiding in the graphical structure of the news propagation on the Twitter social network, that our GNNs could leverage efficiently.
Following our above discussion, we identify two main axes of improvement on this work. First, to improve the generalization performance of our models and limit overfitting, via clever regularization and hyperparameter tuning. Then, our study highlights the current shortcomings of the Twitter15/16 datasets. To properly track the progress made on those, it would be important to have defined a reference train/validate split of the data, and to have a neutral third party hold a *held-out* test set. Twitter datasets are also very small, which prevents the development of complex models. Finally, the user features are now outdated, and it would be interesting to see if it would be possible to build a new version the Twitter datasets, coming with publicly available user features that correspond of the time of the events. Creating such a dataset is not possible today as it would clearly violate Twitter's privacy policy. It would be interesting to see if user features can be safely anonymized to permit the creation of such datasets.

Individual contributions We all equally participated in writing all three reports and evenly throughout the project (see Github insights).

- Arnaud: Exploratory Data Analysis of Graphs, GBDT and LSTM models, SEIZ model.
- Alexandre: Data collection and cleaning and preprocessing, NLP features extraction (BERT), NLP-based models (GNNs).
- Manon: Data loading, GNNs Training and Logging pipeline, Result analysis and visualization, Poster.

References

[1] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[2] Fang Jin et al. "Epidemiological Modeling of News and Rumors on Twitter". In: *Proceedings of the 7th Workshop on Social Network Mining and Analysis*. SNAKDD '13. Chicago, Illinois: ACM, 2013, 8:1–8:9. ISBN: 978-1-4503-2330-7. DOI: 10.1145/2501025.2501027. URL: http://doi.acm.org/10.1145/2501025.2501027.

[3] Yang P. Liu and Yi-fang Brook Wu. "Early Detection of Fake News on Social Media Through Propagation Path Classification with Recurrent and Convolutional Networks". In: *AAAI*. 2018.

[4] Yang Liu and Yi-Fang Brook Wu. "Early detection of fake news on social media through propagation path classification with recurrent and convolutional networks". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[5] Yinhan Liu et al. “Roberta: A robustly optimized bert pretraining approach”. In: *arXiv preprint arXiv:1907.11692* (2019).

[6] Jing Ma, Wei Gao, and Kam-Fai Wong. “Detect Rumors in Microblog Posts Using Propagation Structure via Kernel Learning”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 708–717. DOI: 10.18653/v1/P17-1066. URL: https://www.aclweb.org/anthology/P17-1066 (visited on 10/16/2019).

[7] Jing Ma, Wei Gao, and Kam-Fai Wong. “Detect Rumors on Twitter by Promoting Information Campaigns with Generative Adversarial Learning”. In: *The World Wide Web Conference*. ACM. 2019, pp. 3049–3055.

[8] Jing Ma, Wei Gao, and Kam-Fai Wong. “Rumor detection on twitter with tree-structured recursive neural networks”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 1980–1989.

[9] Jing Ma et al. “Detecting Rumors from Microblogs with Recurrent Neural Networks”. en. In: (), p. 7.

[10] Richard Socher et al. “Parsing natural scenes and natural language with recursive neural networks”. In: *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011, pp. 129–136.

[11] Dehua Wang, Yang Zhang, and Yi Zhao. “LightGBM: An Effective miRNA Classification Method in Breast Cancer Patients”. In: *Proceedings of the 2017 International Conference on Computational Biology and Bioinformatics*. ICCBB 2017. Newark, NJ, USA: ACM, 2017, pp. 7–11. ISBN: 978-1-4503-5322-9. DOI: 10.1145/3155077.3155079. URL: http://doi.acm.org/10.1145/3155077.3155079.