CSE 547: Fake News Detection on Social Media

Anton Lykov
University of Washington
alykov@cs.washington.edu

Sandeep Tiwari
University of Washington
sandy972@uw.edu

Venkata Sai Muktevi
University of Washington
vmuktev1@uw.edu

1 Introduction

Over the past decade, it has been made clear that the spread of disinformation can wreak havoc in society by drastically affecting the political climate, spreading misinformation about contagious viruses, or even sparking major fluctuations in stock markets. Fake news can easily propagate through a multitude of channels and instigate global frenzy within minutes. These are concerning times wherein social media dominates as the primary messenger of news and information across the internet for many people all over the world.

Fake news and disinformation has established itself as a looming presence on social media platforms, and it is critical that we find a way to curb the spread of fake news. We propose to build a model to detect and classify fake news so that it can be dealt with before it reaches consumers on social networks, while generalizing well across different contexts and domains. We will study the various interactions of users on social media with posts on news articles. We scope our data to news content from the domains of Politics and Entertainment.

Several existing methods target fake news detection using supervised models but supervised learning can largely be very domain specific and labelling or finding labelled datasets of fake and real news articles is a tedious procedure. We thought it to be interesting to explore an unsupervised graph-based approach to fake news detection and study the important aspects that help characterize fakenews based on Social Media user behaviours and interactions with news articles. Social networks, such as Twitter, take a substantial presence in our lives, and contain data that can provide insight into how users spread information. The key assumption is that user interactions with news posted on Social Media contains underlying information about whether the news is fake or real.

2 Related Work

The landscape of fake news detection is rather broad. The problem has several angles of approach. One of the most recent comprehensive surveys about this problem is presented in Shu et al. [2017], in which the authors categorize the approaches into *news-content* based and *social context* based, and the latter are in turn categorized into *stance-based* and *propagation-based*.

There is a class of supervised and more notably weakly-supervised methods such as Helmstetter and Paulheim [2018] and Zellers et al. [2020]. Next, there are quite a few unsupervised methods. An example is fake news detection using a graph-based approach suggested in Gangireddy et al. [2020] (the paper that we base our project on); another example is Yang et al. [2019], in which authors develop a generative approach which provides a probabilistic model and uses Gibbs sampling.

Finally, there have even been works suggesting actual software tools that could be used to filter out fake news: Aldwairi and Alwahedi [2018].

We base our investigation on Gangireddy et al. [2020]. In summary, this paper uses social media (Twitter) posts, and articles to construct a graph with edges between posts and articles that are mentioned in these posts. Next, based on the high-level idea that fake news is distributed synchronously (compared to real news) and that there is usually textual similarity of the post contents, the seed labels are given to groups of articles. Then, using article contents similarities and user groupings,

the labels are given to each article in the seed group. Finally, the labels are propagated to all articles using similarity assumptions with between labelled and unlabelled articles.

3 Motivation

Unsupervised fake news detection on social media based on a graph structure is a very novel and interesting approach. However, as we explored the GTUT algorithm deeper, we realized it has limitations. From a high level perspective, these are the main limitations. The first one is that the GTUT method is based on several unmotivated assumptions, such as temporal and textual similarities of articles which may not necessarily hold in reality, and thus not may not be data-driven. For instance, the authors are using textual similarities of the of articles to support the fact that their labels (fake or real) should be the same, which is unmotivated and might be wrong.

With this in mind, we decided to take a fresh look at the problem. We tried alternative methods of doing fake news prediction while staying within the same graph-based setup. Using the user-article graph (and induced graphs, such as article graph). we tried to find hints in the data, in hopes to make our approach more justified.

3.1 GTUT Algorithm: Summary and Pitfalls

The original GTUT algorithm consists of three phases, each of which label all articles based on pre-conceived assumptions about how fake news is propagated across social media. The first phase is primarily based on the notion that most fake news is spread within a social network more synchronously than real news spreads to ensure wide reach and attention to various agendas. Due to the broad nature of this assumption, we only use it as a heuristic to identify candidates for fake and real articles. This seed set is constructed by identifying posting patterns involving similar articles at similar times.

We first construct a bipartite graph $\mathcal{G}$ with nodes for articles and users. An edge is drawn between a user node u and article node a if there exists a post by the user that mentions the article. We then identify all maximal bi-cliques in this graph. The articles of a bi-clique are scored, and then summed to evaluate whether the articles are real or fake. We are then able to generate a seed set of real and fake articles. The second phase consists of label spreading, by using the seed set to label the remaining unlabelled articles in the bi-cliques, based on a predefined similarity metrics considering user similarity, bi-clique similarity, and textual similarity of their posts. The third phase focuses on labelling all articles that reside outside of the bi-cliques, by using a similar metric to implement label spreading based on the articles that were labelled over the first 2 phases.

After implementing this algorithm, we realized there are several weaknesses and pitfalls to the assumptions that were being made. We believed we could improve upon the initial heuristics of the algorithm. The initial assumption that this entire algorithm is based on, is that fake news is generally propagated by different entities in a very short time span. This makes sense intuitively, however, we believed it would not be enough to distinguish it from real articles, and that it could be further elaborated upon with more temporal information. Furthermore, we also believed the use of "textual similarity" using `word2vec` was fairly weak, as real and fake articles could have the same content, syntactically, making it hard to be used as a distinguishing feature.

The pseudo-code for each phase of the algorithm can be found in the Appendix.

4 Our contribution

Our contributions in the work are the following:

1. We debugged and adapted an existing pipeline for obtaining the fake news data set (fak). FakeNewsNet was very slow. We had to rewrite some code and reconfigure the pipeline to run quicker web crawling and data processing in batches. We deployed this on a GCE instance (n2-standard), and experimented with multiple configurations of the FakeNewsNet pipeline.
2. We implemented GTUT as described in the paper, and then adapted it after finding that the original code was left unfinished and did not work properly. We specifically abstracted the

algorithm into each of its phases separately so that we would be able to seamlessly plug in our ideas anywhere in the pipeline.

3. We developed, analyzed, and evaluated two alternative graph-based methods for fake news prediction:
   - The first one is based on the article graph induced by the User-Article graph, where we model similarities between articles based on the graph similarity features such as the number of users sharing both articles, time of sharing, and other such important attributes.
   - The second is the User-Article graph model. Here, we model user credibility score based on the user metadata and the graph structure to predict an article label based on this user credibility score. With this approach, we found both positive and unpromising results.

In the following sections, we discuss each of the contributions in detail after diving into our data and data collection process.

5 Data

5.1 Data Summary

Our data was retrieved using FakeNewsNet [fak]. FakeNewsNet can be described as a multidimensional data repository collection system. It contains initial datasets and a data collection pipeline system for which these initial datasets act as metadata. Implementing this pipeline allows us to retrieve much larger data objects to be used for building our final dataset. The pipeline implements a series of web crawlers and Twitter APIs to collect data. These initial datasets were collected by the creators of FakeNewNet [fak] using web crawlers on the fact-checker websites. Here we will refer to these initial datasets as metadata that was used to collect our final dataset of JSON objects and will explain how they were obtained in the next section.

We have four different labelled metadata sets. Two sets of news articles deemed fake and real by PolitiFact and similarly two from GossipCop. We will talk more about how this metadata was collected in the next section. Table 1 and 2 gives us a summary of the initial datasets. An important point to note here is that there is a lot more data from GossipCop compared to PolitiFact.

News Source	Fake	Real
PolitiFact	432	624
GossipCop	5,323	16,817

Table 1: Number of articles from the metadata.

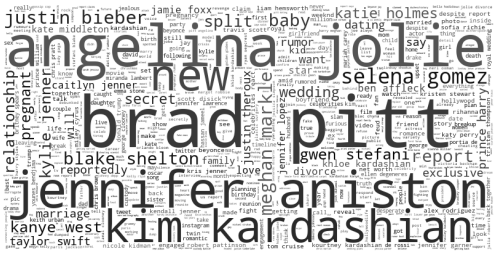
News Source	Fake	Real
PolitiFact	164,892	399,237
GossipCop	519,581	876,967

Table 2: Number of Associated Tweet IDs from the metadata.

To get an idea of what type of news content is in these datasets we generated word clouds of all the headlines of articles from each dataset. This gives us a snapshot of the main topics or themes that the different fact-checkers operated upon. Using the word clouds we can notice the difference in news content being targeted by each fact-checker. GossipCop leans into working with topics related to the entertainment industry, movies, actors, etc. while PolitiFact works with more political news content. This variation is useful for observing the generalizability of our approach to different types of news.

The main data objects that we retrieve after implementing the reconfigured and modified version of FakeNewsNet to construct this dataset are of 3 types: news content, tweets and user profiles. These attributes contribute in different ways to our analyses and models.

To summarize, we gathered data from a total of 23,196 news articles that were scored as real or fake by the fact-checkers PolitiFact and GossipCop. Regarding Tweets, we have data on 564,129



(a) GossipCop Fake News

(b) GossipCop Real News

(c) PolitiFact Fake News

(d) PolitiFact Real News

Figure 1: The word clouds of text from headlines of the "seed" dataset news articles.

tweets from the associated articles checked by PolitiFact and similarly 1,396,548 tweets from articles checked by GossipCop. We have 472,101 user profiles in total. The metadata and news content are publicly available but the Twitter data (tweets and user profiles) has to be crawled using API keys that are only provided after going through due process by Twitter. This data cannot be shared for public viewing.

Table 3 below gives us a summary of the final article data that was collected from each of the four sets of metadata.

News Source	Fake	Real
PolitiFact	408	617
GossipCop	5,150	15,215

Table 3: Number of articles for which content was collected.

An observation here is that the number of articles actually retrieved is not the same as the number of articles originally promised in the metadata for each set. Some of these articles may have been taken down. We found a similar discrepancy in the number of data objects retrieved regarding twitter data as their policies ensure some posts or profiles would be taken down periodically based on suspicious activities or illegal spread of misinformation.

Figure 2 shows a brief description of our data collection procedure for building the different components of our dataset. We will talk more about this in the next section.

5.2 Data Collection

Based on the configuration provided, FakeNewsNet can return several different types of data objects that can be used. To be clear about how we used FakeNewsNet (fak) to collect our data we delve into its inner workings. There are two important points about this collection procedure. Firstly, the metadata was obtained and provided in the FakeNewsNet repository beforehand and the implementation code of the collection procedure for this metadata is not publicly available. Secondly, the final dataset that we obtained is by using the FakeNewsNet system of web-crawlers and Twitter API calls on the metadata. The FakeNewsNet fak system is a publicly accessible data collection pipeline.

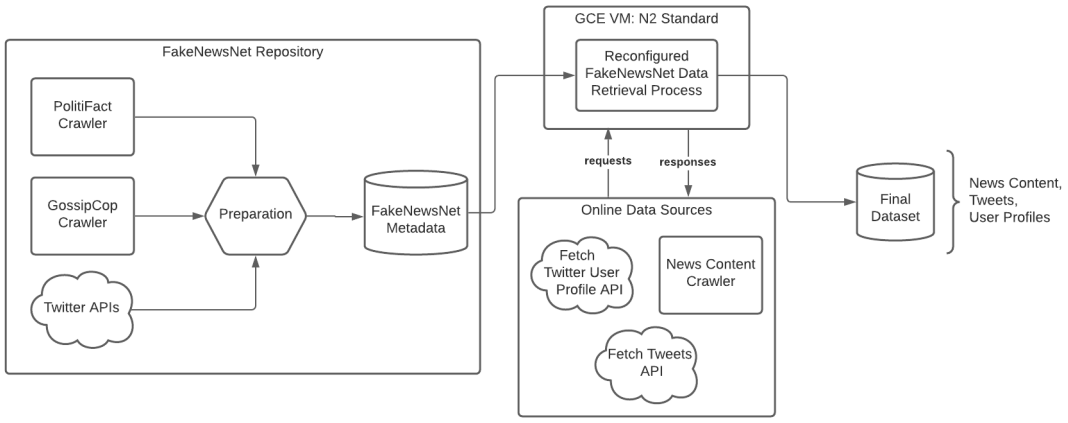


Figure 2: Data collection procedure

5.2.1 The Metadata

The metadata is a collection of news article URLs from two main fact-checking websites namely "PolitiFact" (pol) and "GossipCop" (gos). These websites consist of analysed news articles from various sources that have implemented manual fact-checking techniques on news articles and label news as either real or fake. Their labelling methods are on a scale of how true or false information in the article is. They give a "truth score" to news articles based on what percentage of content in the article is real. This scale is measured differently for each news source. Therefore, here it is considered that partly fake news is also fake news in every case. The assumption is that its important that no false information is shared across social media even if it is just one or two paragraphs containing false information in an article that is untrue. Hence its considered that only if an article is given a complete full "truth score" by the standards of the fact-checker, then it is considered real news. In Figure 2 we can see that the metadata is collected after crawling news articles and their labels from a politifact and gossipcop crawler and then using a preparation phase to combine this data with Twitter IDs.

For each of these news articles we have the associated Tweet IDs of twitter posts that mention the article or that are textually similar to the article's headline. These were obtained using Twitter's Advanced Search API. If any additional URLs were mentioned in the tweets being checked, then queries were based on those URLs were included as well to get more tweet IDs of posts related to the news content being discussed.

Hence, the metadata consists of 4 sets of data. Two sets of Article URLs and associated TweetIDs labelled by PolitiFact as fake and real, along with two sets labelled as fake and real by GossipCop.

5.2.2 The Final Dataset

The metadata is fed into the FakeNewsNet system with the necessary configuration to obtain our final dataset for analysis and modelling. The main data objects we obtained are as follows:

1. News Content: The contents of the news articles crawled from the URLs retrieved as JSON objects. The content is in text format along with metadata like: images, published data, source, url, and so on.

2. Tweets (Twitter Posts): JSON objects obtained using the Twitter API from the tweet IDs. These contain metadata about Tweets which include the following attributes: tweet ID, tweet text, associated user ID, created-at timestamp, language, and coordinates.

3. Twitter User Profile Data: JSON objects also obtained using the Twitter API for all the associated user IDs that interacted with the tweets. These contain metadata about the users such as the following: user ID, user name, screen name, followers count, friends count, listed count, favorites count, location, profile image details and so on.

We modify these data objects as and when necessary for our analysis and modelling into various formats. How these are used will be discussed in later sections.

5.3 Further Exploration

Taking cues from the FakeNewsNet repository construction, we had a concern regarding the presence of bot profiles on social media platforms. We take a deeper look into the user profiles and assess the effects of bots on our data. We used Botometer bot, which takes a Twitter username or ID as input and utilizes various features extracted from the user's profile to obtain a probability score in [0,1], indicating how likely the user is a bot in various aspects. We found that bots are significantly more likely to post tweets related to fake news compared to real users after running it over our dataset. Almost 22% of users involved in fake news are bots, whereas only 9% of users are labelled as bot users for real news.

We also observed the creation times of user accounts based on groups that share fake or real news. We checked if the average age of a profile is different between these fake news sharing groups and real news sharing ones. This gave us more insight into bot users that may be created only for the purpose of spreading fake news. We also observed there was a significant difference between the account creation time of real users versus fake users.

6 Results and Error Analysis

In this section we present the first-principled approach we took. We experimented with the data and tried to see what aspects in the social network structure affect and contribute the most to certain articles being fake or real. We believe this approach is the right way to tackle the problem: rather than start from the unproven underlying assumptions, we look at the data instead. It did not turn out to perform better than the original GTUT score overall, but this was not the goal. The goal was to conduct in-depth experiments in order to understand the meaning of each of the features used in the algorithm.

6.1 User-Article Graph

The User-Article graph from the Gangireddy et al. [2020] captured the our attention the most from the paper. Since social networks graphs usually contain lots of useful information, our main assumption and hope was to extract as much as possible about fake news relying on the bipartite graph of Twitter Users on one side, and Articles tweeted by these users on the other side. We spent most of the time experimenting with this graph as well as its subgraphs and induced graphs. Below we present and discuss our main 3 findings from the User-Article graph.

1. First, we found a somewhat unsurprising but at the same time very interesting result. If we move our focus from classifying articles to classifying users, it turns out that the problem is almost exactly equivalent. To quantify, we looked at the users and labeled users as `FAKE` if they authored **at least 1** fake article, and `REAL` otherwise. Using this approach, the accuracy was $97\%+$! This result shows that the fake news detection on social media could be studied almost equivalently from the perspective of real-fake users, which may give more insight, as number of users is significantly larger than the number of articles.
2. We modeled the credibility scores of the users by looking at the metadata of the users. The main metadata fields were the **followers count**, **friends count**, **verified (bool)**, and **created at**. We explored the data to see if there were any notable differences in statistics among these metadata fields if a user authored fake, real, or both kinds of articles. See Figure 3 for statistical difference based on the **verified** status. Note that total number of user is also different in the two graphs.

   This status turned out to be a helpful metric. Among the models based on the metadata, we managed to get a relatively high precision – **0.74** – by looking at the user's verified status and labeling article as true if any user who authored it is verified (but overall accuracy only of about 0.65).

   Also, it might be interesting to look at the subgraph consisting only of the verified users in Figure 4. The fake articles are labeled red, and real – green. Note how the number of fake articles having out-edges is very small, except for one article, which we found very surprising:

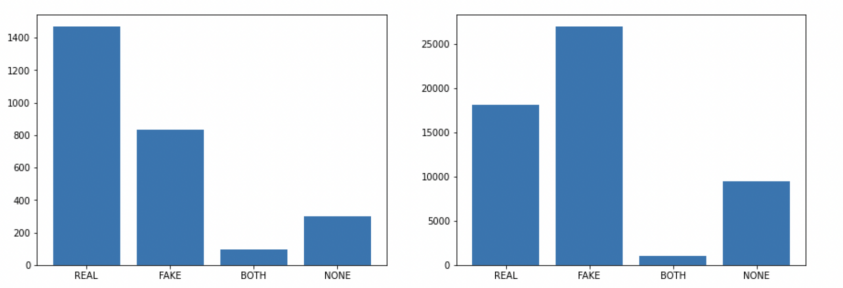
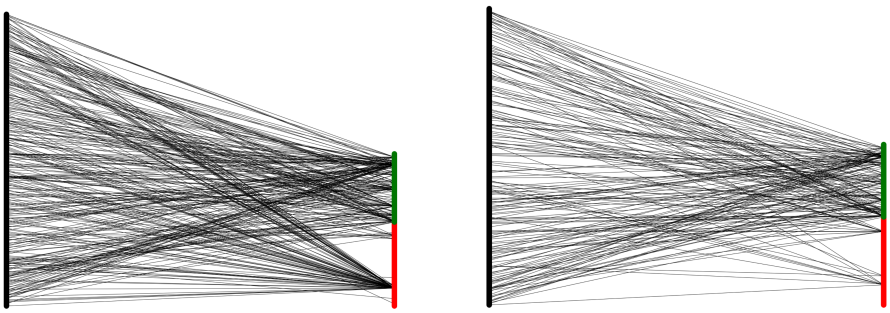


Figure 3: Separation of labels based on whether a user is verified (left) or not (right). Politifact sample.

`https://www.house.gov/the-house-explained/the-legislative-process`. If we remove that outlier, the number of edges from fake-articles is very small.



(a) Including an outlier article

(b) Excluding an outlier article

Figure 4: Verified users only; fake (red) and real (green) articles. Politifact sample

3. We looked at a graph induced by the users, where the weights between users U_1 and U_2 is given by $|\{A : A \text{ mentioned by both } U_1, U_2\}|$. We ran several community detection algorithms in this user subgraph, and discovered interesting results. The communities found by the Clauset-Newman-Moore greedy modularity maximization algorithm split the users into communities such that in each community the users belonged to either turned out to all have either fake or real label ($90\%+$ accuracy). We think this very promising, but at the same time there's no reliable way to label each community is true as either true or false.

6.2 Article Graph

Another approach we took was to simply assess the manner in which articles were propagated with respect to one another. We modeled this in the form of an article graph, where edges between articles were drawn based on if at least one user shared both articles. Based on this initial heuristic alone, we were able to generate the following graph:

Figure 5 is color-coded according to the ground-truth labels, where the blue nodes correspond to legitimate articles, whereas the red nodes correspond to fake articles. Given only the information about which articles are shared by the same user, there are several very interesting observations we can make regarding the structure of this graph. We can see that all real articles are clustered fairly tightly together, with the fake article nodes residing primarily on the outskirts of the graph. There is some noise, but it provided us with inspiration to attempt to see if we could formulate some sort of weighting system that would allow us to perform a min-cut bisection to partition the real and fake articles in a reliable way. When running the min-cut bisection algorithm on the unweighted graph, we obtained an accuracy score of about 0.6315.

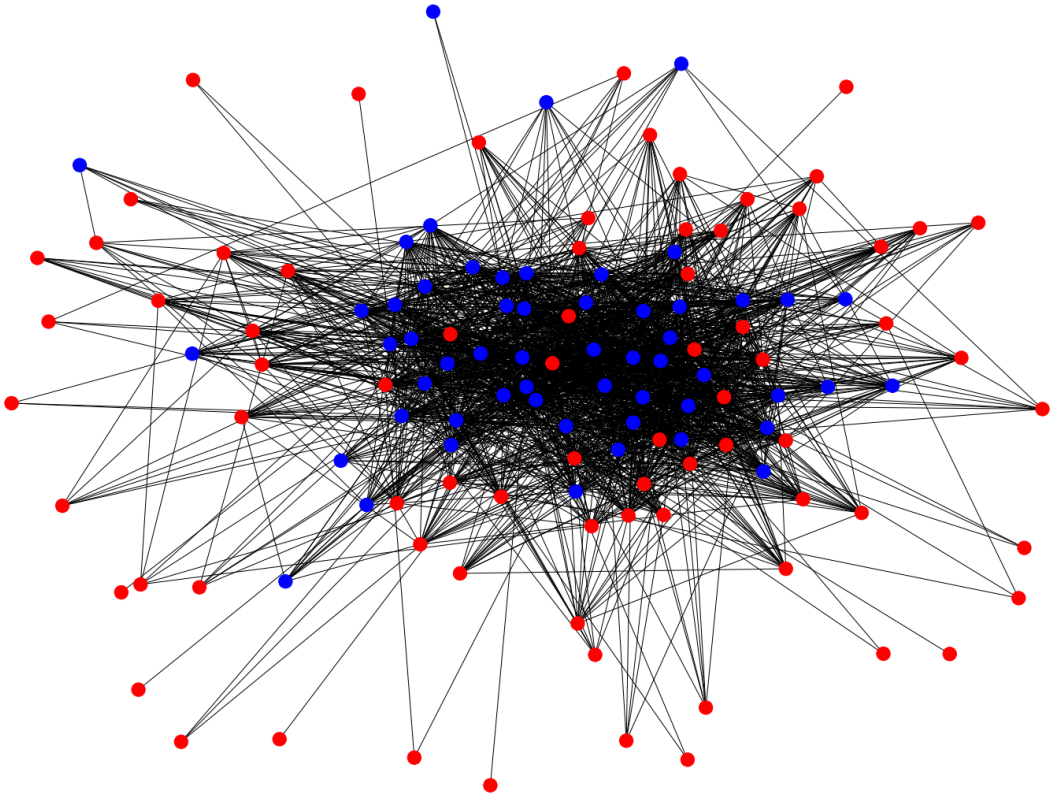


Figure 5: Randomly sampled subgraph of the article graph

We attempted to utilize the article and user metadata to amplify the inner-connectedness of the article labels. So, we wanted the weights of edges between articles of the same label to be very high, and the edges between articles of different labels to be low. Intuitively, this would allow a min-cut algorithm to effectively partition the articles. We utilized an additive formula for establishing a similarity metric as it produced the best performance, as it would cause highly similar articles to be close together. We took into consideration timestamps of the tweets mentioning the articles, and similarity of the text of the articles. We also attempted to leverage user metadata; since each edge corresponded to users sharing both articles, we examined Twitter verified status, number of followers, number of tweets, and account age. The intuition behind using these user features, was that verified users with more followers and a broader reach are extremely likely to be reliable than others. So we generated highly weighted edges that correspond to users with these particular attributes. Our weighting metric read as

$$Weight(A, A') = \frac{\sum_{u \in \mathcal{U}} (\text{\# of followers})_u}{|Users(A) \cap Users(A')|}$$

Unfortunately, we did not find this approach to be fruitful, as the trends of the features we were taking per user are very transient, and do not demonstrate a consistent pattern of behavior, specific to sharing fake news. So it became difficult formulating a weighting metric that would embed this information, while creating separation between the different labels of articles.

6.3 Modified GTUT

Using the results of our findings outlined in **Section 6.1**, we attempted to label the users as reliable or not, based solely off the user metadata. This would provide us with some seed data upon which we could then label the articles. Our approach was to compute a *reliability score* using the user metadata, `verified status`, `followers count`, `tweets count`, `favourites count`, and `account age`. Once again, we strongly took into consideration an *influence factor* that described the influence a user

had in spreading information on Twitter. Our heuristic is that users are more likely to be reliable if (i) *they are verified on Twitter*, (ii) *they have a large following*, or (iii) *their Twitter account was created a while ago, and has been regularly active since then (i.e. not a bot recently made for spreading misinformation)*. Naturally, verified status is a binary feature, which is why it was incorporated to compute a "reliability score". This reliability score R reads as

$$R(U) = 1 - (1 - I_U) \cdot \left(\frac{1}{\text{followers count for user } U} \right) \cdot \left(1 - min \left(\frac{\text{account age}}{\text{tweets count}}, \frac{\text{tweets count}}{\text{account age}} \right) \right)$$

where I_U is an indicator variable for whether or not a user is verified.

This reliability score is between 0 and 1, where 0 indicates a low reliability, and 1 indicates a high reliability. We tuned our hyperparameter minimum threshold τ, where a user is labeled as reliable (1) if their $R_U >= \tau$. We computed best accuracy by comparing our predictions to the ground truth labels, where the computations of the labels are defined in point 1 of **Section 6.1**. However, we found using a threshold of $\tau = 0.65$ to produce a best accuracy of only 0.5546, which deterred us in continuing with this approach. We attribute the fairly weak predictive power of our reliability score to the relatively small feature space. While the Twitter API provided us with a large amount of metadata about each user, we found that the number of features that were relevant to describing a user's Twitter activity was relatively low.

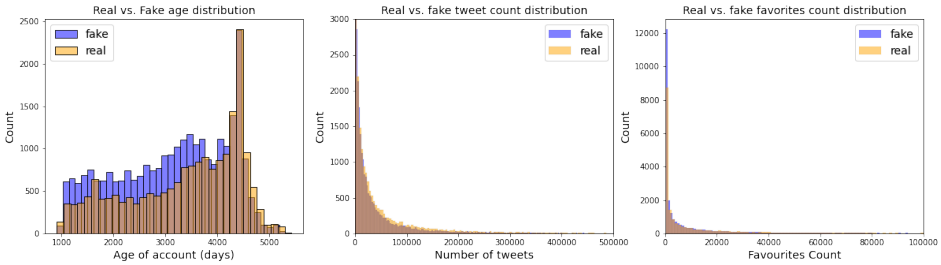


Figure 6: Distribution of features based on user ground truth "label"

After further analysis, Figure 6 shows that there is little to no separation between the labels based on any feature in particular, making it very difficult to accurately label the users, given how small the feature space is. This is why we instead decided to revert to labelling the users based off the article labels. However, in order to maintain the unsupervised nature of the algorithm, we decided to label the users based off the seed labels produced in phase 1 of the GTUT algorithm. Once we obtained the labels of the users using the seed data, we hoped to propagate the label spreading to other unlabelled articles shared by those users that had shared at least 1 fake article (according to the seed labels produced in phase 1).

As previously mentioned, the structure of our implementation of the GTUT algorithm was highly modularized so that we could seamlessly plug in any of our adaptations to the algorithm. After Phase 1, where the original GTUT algorithm generates a seed set of labels, we introduce a Phase 1.5, where we label the users, and then label all other articles shared by those users if assigned a "fake" label. These article labels are then passed into Phase 2. The motivation behind this is to add an extra dimension to the article labelling process (using article metadata), while speeding up the label spreading process as well by leveraging the high correlation between user and article labels.

Implementation	Time (minutes)
Without Phase 1.5	58.935
With Phase 1.5	39.987

Table 4: Timing comparison for GTUT with and without Phase 1.5

As expected, we found a noticeable speedup in the overall runtime of the algorithm, despite adding an additional intermediary step in between Phases 1 and 2. This stemmed from the relatively minimal

time to label the users, and propagate that information the the remaining unlabelled article, ultimately resulting in less computation in the following phases.

However, we did actually find that implementing the GTUT with the inclusion of this step did not change the results at all. So we solely saw a speedup in runtime, with little to no change in performance. While we were unable to accomplish the original task at hand, we did find an advantage to one of our approaches that should strongly be considered in future variations of this algorithm.

6.4 Analysis and errors

The dataset we worked with is a snapshot of the real world data. We believe that the main potential discrepancies between our work and real world is that the data might not be representative enough (it contains only a small fraction of all the news posted every day on many social media platforms). Further, we also obviously rely on the correctness of the labeled data, which we cannot check completely. Other than that, we did not do any approximations and other hidden assumptions.

7 Challenges and Issues

Regarding data collection the main challenge was that the Twitter API had limits on the requests and the size of requests per day. We had several hundred thousands of requests to be made to completely collect our data. This led to the data collection extending for days at a stretch. Using GCP came very handy in this situation by allowing us to offload working our resources to the google cloud. However, it didn't speed up the process as the issue was with the limits of the Twitter API and we couldn't efficiently implement the multiprocessing features of FakeNewsNet without more than two Twitter API keys. Working with private data from social media websites like Twitter has many limitations.

The original source code for GTUT was incomplete and undocumented. This required us to provide our own implementation of the GTUT algorithm and also refactor the code to follow the algorithm more closely. This not only required us to build our own implementation, but also make it modularized so that it would be easy to plug our adaptations into it.

Furthermore, we ran into several time complexity issues with making the textual comparisons. Using `word2vec`, even with a pre-trained model, turned out to be very slow when performing pairwise comparison of article and post texts across the different phases. We had to circumvent the use of this component in the overall model in lieu of the aforementioned proposed adaptations.

8 Future Work

One thing we wanted to explore was the use a network of users from the dataset, by leveraging the follower and following metadata that we could have obtained from the Twitter API. One of our initial ideas was to generate a user network and perform a trust rank, to evaluate reliability of the different users available to us. This would have been able to play into our proposed Phase 1.5, which would allow us to already have pre-computed reliability scores for each user.

Another direction we would like to look at is to perform a more robust NLP analysis on the textual information at hand, by analyzing the syntactical and semantic structure at a more granular level. We believed that if an article uses language to suggest that its speculating or not completely grounded in truth, it should be more scrutinized. We also speculate that there would be a high correlation between using speculative language and being unreliable.

Finally, a promising research direction is to see how adding supervision improves the models like GTUT and models that we proposed. Specifically, one could use supervision in place of the seed set, or to label the communities detected in the User-graph. Comparing how variable degrees of supervision perform could eliminate the questions and doubts we have about some of the assumptions of the GTUT being unmotivated.

References

Botometer repository. https://github.com/IUNetSci/botometer-python.

Fakenewsnet repository. https://github.com/KaiDMML/FakeNewsNet.

Gossipcop. https://www.gossipcop.com/.

Politifact. https://www.politifact.com/.

M. Aldwairi and A. Alwahedi. Detecting fake news in social media networks. *Procedia Computer Science*, 141:215–222, 2018. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2018.10.171. URL https://www.sciencedirect.com/science/article/pii/S1877050918318210. The 9th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2018) / The 8th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2018) / Affiliated Workshops.

S. C. R. Gangireddy, D. P, C. Long, and T. Chakraborty. Unsupervised fake news detection: A graph-based approach. In *Proceedings of the 31st ACM Conference on Hypertext and Social Media*, HT '20, page 75–83, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370981. doi: 10.1145/3372923.3404783. URL https://doi.org/10.1145/3372923.3404783.

S. Helmstetter and H. Paulheim. Weakly supervised learning for fake news detection on twitter. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 274–277, 2018. doi: 10.1109/ASONAM.2018.8508520.

K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu. Fake news detection on social media: A data mining perspective. 2017.

S. Yang, K. Shu, S. Wang, R. Gu, F. Wu, and H. Liu. Unsupervised fake news detection on social media: A generative approach. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33 (01):5644–5651, Jul. 2019. doi: 10.1609/aaai.v33i01.33015644. URL https://ojs.aaai.org/index.php/AAAI/article/view/4508.

R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi. Defending against neural fake news, 2020.

Appendix

GTUT Algorithm

Algorithm 1 GTUT Phase 1: Label Seeding using Bi-cliques

$\mathcal{U} \leftarrow$ set of user ids
$\mathcal{A} \leftarrow$ set of user articles
$\mathcal{P} \leftarrow$ set of user posts
$\mathcal{G} \leftarrow$ empty bipartite graph with nodes as users and articles
$\lambda \leftarrow 0.5$
$\tau_{hi} \leftarrow$ minimum threshold for TT-score for fake labels
$\tau_{lo} \leftarrow$ maximum threshold for TT-score for legitimate labels
for post in $\mathcal{P}$ **do**
    for user U, article A in post **do**
        create edge $U \rightarrow A$ in $\mathcal{G}$

identify all maximal bicliques as a set $\mathcal{B}$
for biclique B in $\mathcal{B}$ **do**
    $B_A \leftarrow$ set of articles in biclique B
    for article A in B_A **do**
        $Temporal(A, B) = max\{1 - \frac{BAS(A,B)}{T_{max}}, 0\}$
        $Textual(A, B) = \frac{\sum_{P_x, P_y \in Posts(A,B)} sim(rep(c_x), rep(c_y))}{(|Posts(A,B)| \times |Posts(A,B) - 1|)}$

    $Temporal(B) = \frac{\sum_{A \in B_A} Temporal(A,B)}{|B_A|}$

    $Textual(B) = \frac{\sum_{A \in B_A} Textual(A,B)}{|B_A|}$

    $TTScore(B) = \lambda \times Temporal(B) + (1 - \lambda) \times Textual(B)$

$T_{seed} \leftarrow$ seed set of legitimate articles
$F_{seed} \leftarrow$ seed set of fake articles
for article A in $\mathcal{A}$ **do**
    $TTScore(A) = \frac{\sum_{B \in \mathcal{B}} TTScore(B)}{|BiCliques(A,B)|}$
    if $TTScore(A) \geq \tau_{hi}$ **then**
        add A to F_{seed}
    else if $TTScore(A) \leq \tau_{lo}$ **then**
        add A to T_{seed}

Algorithm 2 GTUT Phase 2: Label Spreading within Bi-cliques

$\mathcal{G}' \leftarrow$ empty graph of articles within bicliques
$G_A \leftarrow$ set of articles within bicliques
for article A in G_A **do**
    for article A' in G_A **do**
        $Weight(E(A, A')) = \alpha \times \frac{|Bicliques(A) \cap Bicliques(A')|}{|Bicliques(A) \cup Bicliques(A')|} + \beta \times \frac{|Users(A) \cap Users(A')|}{|Users(A) \cup Users(A')|} + (1 - \alpha - \beta) \times Sim(A, A')$
        create edge $A' \rightarrow A$ in $\mathcal{G}'$ with weight $Weight(E(A, A'))$
for all unlabelled articles **do**
    label article with same label as labelled article that shares largest edge weight in $\mathcal{G}'$

Algorithm 3 GTUT Phase 3: Full Dataset Labelling

$\mathcal{Y} \leftarrow$ set of articles outside of $\mathcal{B}$
$\mathcal{G}_{\mathcal{Y}} \leftarrow$ empty graph of articles not within bi-cliques
for article A in $\mathcal{Y}$ **do**
    for article A' in $\mathcal{G}_{\mathcal{A}}$ **do**
        $Weight(E(A, A')) = \gamma \times \frac{|Users(A) \cap Users(A')|}{|Users(A) \cup Users(A')|} + (1 - \gamma) \times Sim(A, A')$
        create edge $A' \rightarrow A$ in $\mathcal{G}_{\mathcal{Y}}$ with weight $Weight(E(A, A'))$
for article A in $\mathcal{Y}$ **do**
    label article with same label as labelled article that shares largest edge weight of A in $\mathcal{G}_{\mathcal{Y}}$