Hanoi University of Science and Technology

School of Information and Communication Technology

# Final Report

**Subject: Graph based Fake News Detection**

**Course: Web Mining**

*Group 10:*

| | |
|---|---|
| Nguyen Ngoc Khanh | (20204915) |
| Tran Ngoc Khanh | (20200326) |
| Tran Quoc Khanh | (20204881) |
| Tran Cat Khanh | (20204916) |
| Ho Minh Khoi | (20204917) |

*Instructors:*
Nguyen Kiem Hieu PhD.

Dec. 2023

## Contents

## 1 Introduction

In the age of technology, people are slowly being overwhelmed with a massive amount of information. While they provide us with a lot of new knowledge about the world, many things that we see on the Internet may not be correct. The most straight forward way one can use to determine whether the information they see is false or not is fact-checking. However, not everybody has enough domain knowledge, common knowledge, and time to get the work done. In contrast to human's inability and limited resource, the amount of news generated daily is getting bigger and bigger through the time. According to Techjury [10], over 650 million tweets posted per day in 2022, which means that even if only 1% of them contain false information, the amount we have to get rid of is still a relatively large number. Thus, the task of detecting fake news automatically is becoming of importance and is an interesting subject that attracts the attention of many researchers.
In our project, we experiment a variety of approaches themed graph-based modelling on FakeNewsNet dataset [13, 12, 11] and UPFD dataset [4]. Results shows that we are able to reach a 90% accuracy for this task, which is a promising number and leaves many room for future researches. Our codes are available at https://github.com/hmkhoi2701/fake_news_detection

## 2 Dataset and data collection

UPFD (**U**ser **P**reference-aware **F**ake News **D**etection) dataset [4] is a large scale dataset related to fake news and interactions on Twitter. UPFD is built upon FakeNewsNet [13, 12, 11]. Figure 3 depicts the data integration pipeline for FakeNewsNet.

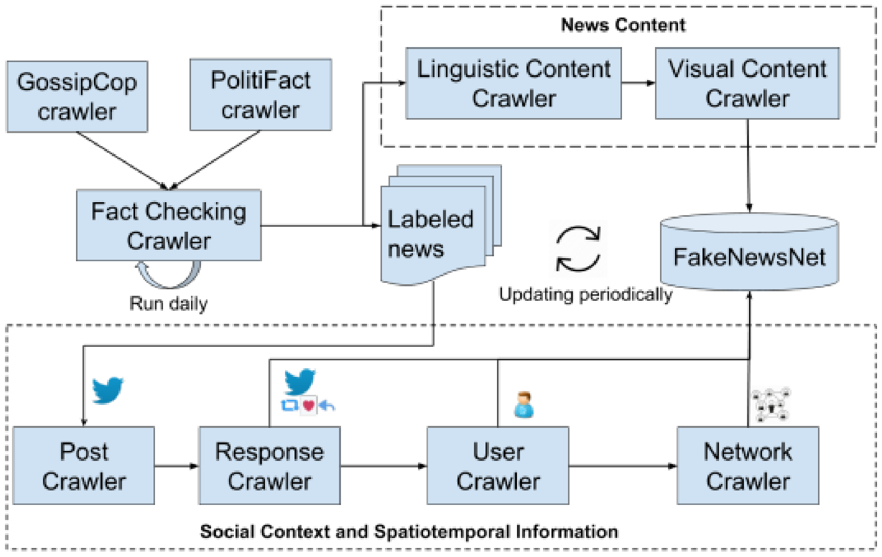

Figure 1: Data integration of FakeNewsNet. Source: [11]

In FakeNewsNet [13, 12, 11], fact-checking websites including PolitiFact [1] and GossipCop [2] are ultilized to provide fact-checking results as label. While the whole process seems expensive and

[1]https://www.politifact.com/
[2]no longer working

complicated, most part of the data are not allowed to be publicized due to Twitter's policy. Therefore remaining data felds only include the news' URL, title and list of IDs of tweets that shares the news. The list of IDs are useful in some way. Firstly, the length of the list highly correlates with the number of users that interacts with the news. Furthermore, from the tweet ID, we can extract more textual information which helps enrich our data. Finally, with any observed tweet, one can get the information of its author, including the number of posts, user interactions and even geographical information, all of which reflects how trustworthy the account is.

| Subset | Label | No. news | Max tweets | Collected |
|---|---|---|---|---|
| GossipCop | Fake | 5323 | 2568 | 4228 |
| | Real | 16817 | 2082 | 12728 |
| PolitiFact | Fake | 432 | 29060 | 270 |
| | Real | 624 | 27377 | 408 |

Table 1: FakeNewsNet dataset attributes

Despite the fact that FakeNewsNet provides sufficient meaningful information as stated above, they cannot bring extra information into the dataset files. Therefore, we have to integrate our tools to crawl the data from source. We used Python's trafilatura [3] library to scrape news from source to enrich our data. While some data is recoverable, many has deprecated URL which makes it impossible to get the text from source. Table 1 shows some analytical attributes of the FakeNewsNet dataset.
From available article titles of FakeNewsNet, we visualized a word cloud in figure 2. As seen in the figure, our dataset contains mostly news about celebrities.

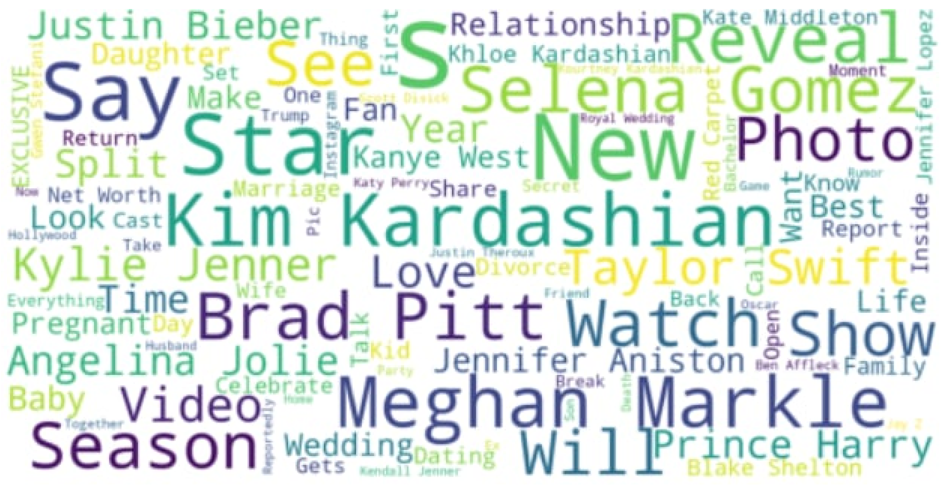

Figure 2: FakeNewsNet title word cloud

In UPFD [4], the authors represent a piece of news and its tweets and retweets in terms of a tree-structured graph, where the root node represent the news article and leaf nodes represent users that tweeted or retweeted the article. Consequently, news-user edges will represent direct tweets from the news source and user-user edges. Moreover, each node can contain 1 of 4 types of attributes:

1. *profile*: A 10-dimensional profile feature. This vector is crawled using Tweepy [4]. It contains

[3] https://github.com/adbar/trafilatura
[4] https://github.com/tweepy/tweepy

verification status, geographical status (enabled), some analytical attributes of the aacount such as number of followers and following accounts, number of posts and media, creation time, and length of account name and profile description.

2. *spacy*: A 300-dimensional feature composed of Twitter user historical tweets encoded by the spaCy [5] word2vec encoder.
3. *bert*: A 768-dimensional composed of Twitter user historical tweets encoded by Jina's bert-at-service [15]
4. *content*: A 310-dimensional feature, concatenation of *profile* and *spacy*.

Finally, each graph will be labled 0 or 1 corresponding to the news being real or fake.
We can visualize some of the news graph as follow and the statistics of the dataset is shown in the table below

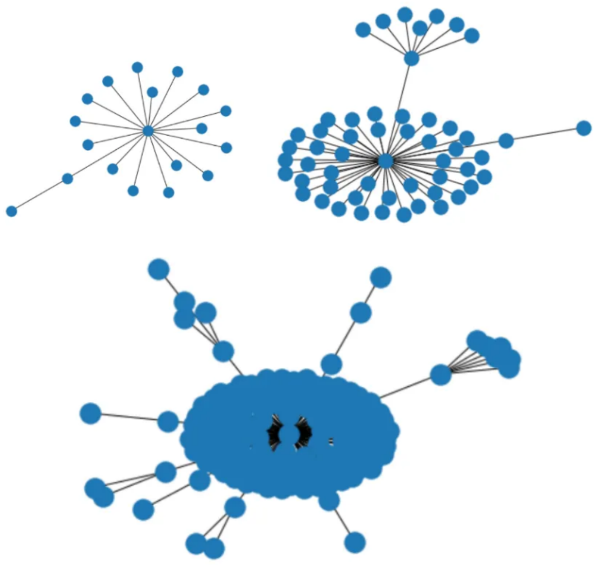

Figure 3: Example visualization of graphs from UPFD [4]

| Subset | No. graph | Real/fake | No.nodes | Train | Val | Test |
|---|---|---|---|---|---|---|
| GossipCop | 5464 | 50:50 | 314,262 | 1092 | 546 | 3826 |
| PolitiFact | 314 | 50:50 | 41,054 | 62 | 31 | 221 |

Table 2: UPFD dataset attributes

[5] https://github.com/explosion/spaCy

# 3 Methodologies

## 3.1 Baseline: NLP-based fake news detection

### 3.1.1 Source-based

For this section, we detect fake news purely based on the publishing source. From the labeled news, we parse the URL of the sources, and extracted the list of websites that uploaded fake news or real news. In total, from FakeNewsNet, we get 1511 real news sources, 553 fake news sources and 425 sources that may share both real news and fake news. For this type of sources, we adopt embedding methods in section 3.1.2 to help classify the news piece based on the title.

### 3.1.2 Content-based

In this section, integrity of news pieces will be evaluated using 3 levels of content: Title, Abstract, and Full Text. First thing we do is to preprocess the text, with the following steps:

- Lowercasing
- Punctation removal
- Number removal
- Stop word removal
- Stemming and Lemmatization using nltk

Finally, the preprocessed contents will be tokenized by TF-IDF with Random Forest classifier [2] or BERT [3].

## 3.2 Graph-based fake news detection

### 3.2.1 Graph representation overview [1]

Traditional machine learning models for graphs usually extract node-level, link-level and graph-level features and then forwarding them through a model that map features to labels. With the introduction of graph representation, the cost of feature engineering for graphs is significantly reduced. Instea, one only have to map nodes into an embedding space with the goal to match similar nodes (or to maximize their dot product). Graph representation methods has moved from fully representative such as adjacency matrix to embedding methods such as DeepWalk[9] and node2vec [5]. In short, DeepWalk ultilizes the idea similar to skip-grams by using random walks to find neighbors (similar to context), while node2vec add two extra parameters, P (return parameter) and Q (in-out parameter) to adjust random walks. While node2vec-family is an effective way to represent a graph, they have some disadvantages, such as lacking ability to captures internal information of nodes or unable to resolve updates in the graph, thus become unsuitable for our problem. On the other hand, graph neural networks seem to be a good choice of model.

### 3.2.2 Graph neural networks

Graph neural networks (GNNs) are a type of deep learning architecture specifically designed to work with graph-structured data. Unlike traditional CNNs that excel at processing grid-like data such as images, GNNs can capture the intricate relationships and interactions between nodes and edges in a graph. GNNs consist of multiple permutation equivariant/invariant functions, where each is a layer. Every layer includes:

- **Message**: Each node will create a message, which will be sent to other nodes later.
- **Aggregation**: A node will aggregate the messages from its neighbors.

The basic GNN can be formulated as follows:

$$h_v^0 = x_v$$
$$h_v^{l+1} = \sigma(W_l \textstyle\sum_{u \in N_v} \frac{h_u^l}{|N_v|} + B_l h_v^l)$$
$$z_v = h_v^L$$

In detail, the initial embedding equals to the node features, while the embedding at layer $l$ equals an activated weighted sum of average of neighbors' previous embedding and the previous embedding of the node itself. This is repeated until the final layer to yield the output. $W_l$ and $B_l$ will be the trainable weight matrices that GNNs will learn

### 3.2.3 A different representation

Apart from a parent-child graph representation as in figure 3, we can also use a network of news content only. Starting with the root node of each news piece in UPFD, we add each to the graph, and link them by *profile* field with a distance below certain threshold $\tau$. We name this data partition **UPFD-Root**. An example is portrayed in figure 4.

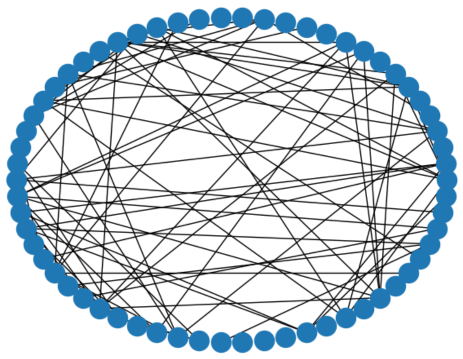

Figure 4: UPFD-Root-Politifact with $\tau$=0.05

### 3.2.4 Methods

Our methods are described in figure 5.
From the graph, we obtain an embedding using a GNN. After that, we use ReLU and batchwise global maxpooling and a linear layer to get a two-dimensional output, representing probability of

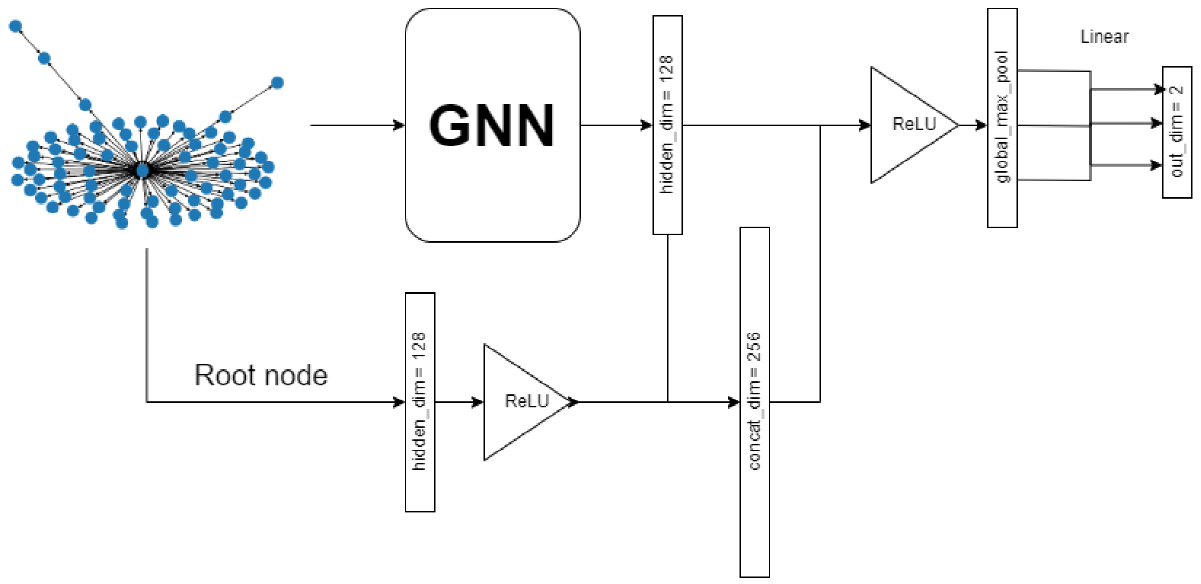

Figure 5: Proposed GNN for UPFD

each class (real or fake news). In case of root node concatenation, the root node embedding will be linearly projected and get concatenated with the output of the GNN to get a 256-dimensional vector, before linear projecting back to a 128-dimensional vector. In particular, we chose 3 types of GNN for experiment.

**GCN (Graph Convolutional Networks)** [8]

The whole basic idea of GNNs can be rewritten in GCN. One extra thing to note is that, we may see GCN written in the form of:

$$h_v^{l+1} = \sigma(W_l \sum_{u \in N_v} \frac{h_u^l}{|N_v|})$$

This is because the input graph is assumed to have self-edges that are included in the summation.

**SAGE (SAmple and aggreGatE)**[6]

In GraphSAGE, the layer output, $h_v^l$ is written as:

$$h_v^l = \sigma(W_l * CONCAT(h_v^{l-1}, AGG(h_u^{l-1}))$$

This can be seen as a two stage aggregation, with the first aggregation of the neighbors resulting in a immediate result (AGG), and the second aggregation acts over the node itself. Multiple aggregation functions are up to use, including mean aggregator, pooling aggregator and LSTM aggregator. Compared to GCN, SAGE efficiently samples a fixed-size subset of neighbors, improving scalability and reducing memory costs. Also, by using sampling neighborhood, SAGE is inductive because it can predict the label of unseen nodes.

**GAT (Graph Attention Networks)**[14]

$$h_v^{l+1} = \sigma(W_l \sum_{u \in N_v} \alpha_{vu} W_l h_u^l)$$

Unlike standard GCNs that treat all neighbors equally, GATs use an "attention mechanism" to selectively focus on the most informative neighbors for each node. This allows them to prioritize important connections and avoid getting bogged down by noisy or irrelevant information. GATs assign "scores" to each edge, reflecting the importance of the connection between two nodes. These

scores are based on the features of both nodes and how well they complement each other. This ensures that high-scoring neighbors have a greater influence on the node's updated representation. Moreover, GATs are computationally efficient since they allows parallelization accross all edges.
Intuitively, let $a$ be a **attention mechanism**, we can calculate the attention score between a node and its neighbor as follow:

$$e_{vu} = a(W_l h_u^{l-1}, W_l h_v^{l-1})$$
$$\alpha_{vu} = \frac{exp(e_{vu})}{\sum_{k \in N(v)} exp(e_{vk})}$$

## 4 Results and analysis

### 4.1 Experiment setup

#### 4.1.1 FakeNewsNet

We evaluate the robustness of the model and quality of the dataset using different splitting methods:

- 80% training - 20% testing
- 50% training - 50% testing
- 20% training - 80% testing

Our experiments shows that Random Forest Classifier with 100 estimators performs slightly better than AdaBoost. We then compare the best result of TF-IDF with BERT-base finetune in 10 epochs with batch size 8.

#### 4.1.2 UPFD & UPFD-Root

Apart from the provided split, we also mimic what we have done with FakeNewsNet to evaluate the robustness of the model and the generalization level of the dataset: full training on GossipCop/PolitiFact and testing on the other dataset
For training UPFD, we use a batch size of 128, Adam optimizer [7] with learning rate 1e-3 and 0.01 weight decay and NLL loss within 100 epochs, whereas for UPFD-Root we use single batch within 500 epochs and BCE loss.

### 4.2 Results

#### 4.2.1 FakeNewsNet

By using **source-based** fake news detection, we discovered 511 real news sources, 553 fake news sources and 425 sources that may share both real news and fake news. However, in detail, sources that may share both real news and fake news mostly because they share multiple news. In 22866 news piece, we can ensure 4484 real news and 1182 fake news, while 17200 news remain ambiguous. With the adoptation of **content-based** fake news detection methods, we get the following results in table 3

| **Model** | **Content-based only** | | **Content&source-based** | |
|---|---|---|---|---|
| | Accuracy | F1 | Accuracy | F1 |
| TF-IDF | 83.22% | 0.8996 | 87.79% | 0.9259 |
| BERT | 83.59% | 0.9003 | 90.99% | 0.9487 |

Table 3: Baseline results - without graphs

### 4.2.2 UPFD

Table 4 and 5 shows the result of the methods of GNN that we used for UPFD dataset.

| **Feature** | **GCN** | **GAT** | **SAGE** |
|---|---|---|---|
| profile | 75.11% | 74.66% | 76.47% |
| spacy | 80.54% | 79.19% | 79.64% |
| bert | 82.35% | 82.35% | 81.90% |
| content | 83.71% | **90.50%** | 87.78% |

Table 4: Accuracy of models and features in UPFD_PolitiFact

| **Feature** | **GCN** | **GAT** | **SAGE** |
|---|---|---|---|
| profile | 89.15% | 91.22% | 92.05% |
| spacy | 96.26% | 95.82% | 96.73% |
| bert | 96.52% | 96.79% | 96.60% |
| content | 95.32% | **97.07%** | 96.68% |

Table 5: Accuracy of models and features in UPFD_GossipCop

We can see that the **content** feature is rich in information, thus we use it with GATConv to examine the robustness of models and dataset in table 6.

| **Train subset** | **Train acc** | **Val acc** | **Test acc** |
|---|---|---|---|
| GossipCop | 97.62% | 96.89% | 50.68% |
| PolitiFact | 98.39% | 80.65% | 43.18% |
| Combined | 94.80% | 93.41% | 94.44% |

Table 6: Inference accuracy

Apart from that, we also observe quite a similar result with and without concatenation of the root node. That being said, a concatenated network is unnecessary for our dataset.

### 4.2.3 UPFD-Root

Using GNNs, the nodes embedding can be projected into a space where the news node are much closer to each other compared to fake news nodes as in figure 6.

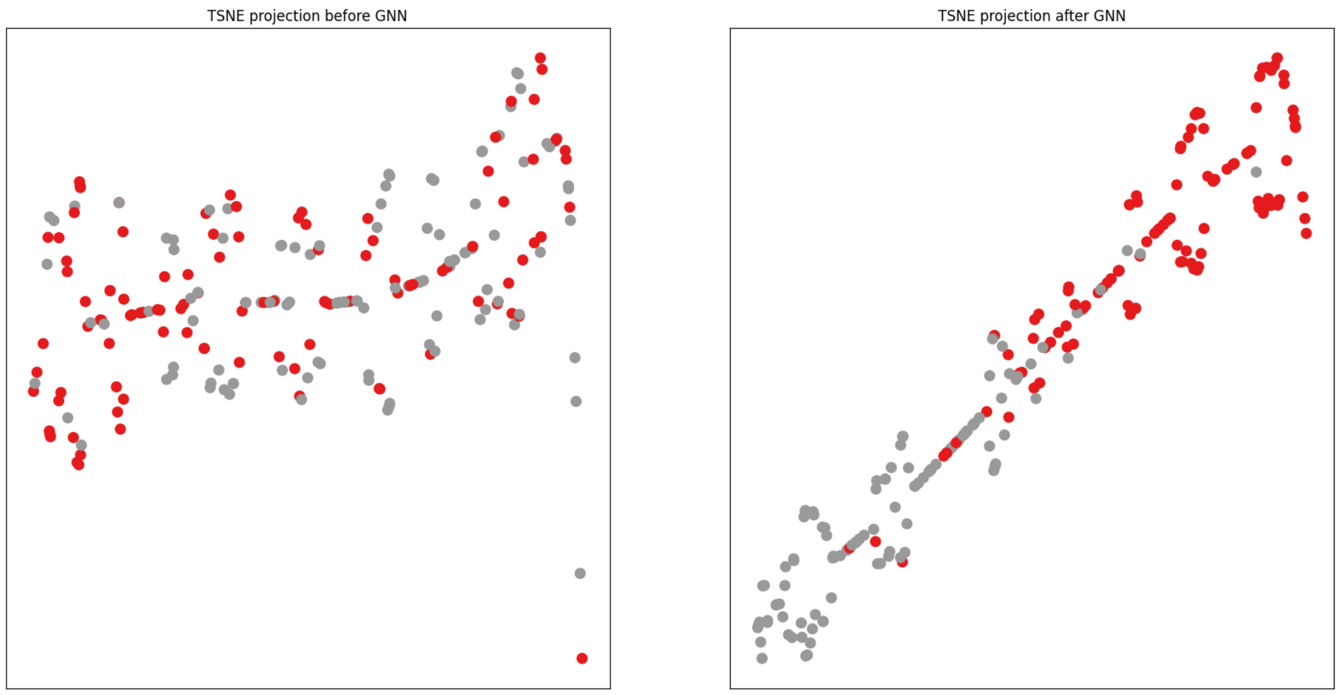

Figure 6: Visualization of node embeddings on 2D space

With initial embedded values, we can use kNN to classify news feed. Using train splits as training set, we obtain the results in table 7. We conclude that k=7,8,9 are the sub-optimal values for kNN.

| **Feature** | **profile** | **spacy** | **bert** | **content** |
|---|---|---|---|---|
| k=1 | 65.16% | 71.04% | 76.02% | 69.68% |
| k=2 | 62.90% | 63.80% | 72.85% | 62.44% |
| k=3 | 71.49% | 72.40% | 71.49% | 67.87% |
| k=4 | 71.49% | 71.49% | 74.21% | 65.15% |
| k=5 | 73.30% | 71.49% | 74.21% | 70.59% |
| k=6 | 71.94% | 73.30% | 75.57% | 67.42% |
| k=7 | 74.21% | 73.76% | 75.11% | 72.85% |
| k=8 | 72.85% | 72.85% | 77.82% | 70.13% |
| k=9 | 73.76% | 71.94% | 72.85% | 75.56% |
| k=10 | 72.85% | 69.23% | 76.47% | 71.49% |

Table 7: kNN tuning results (accuracy) in UPFD-Root

We once again, measure the accuracy on test split of features, using **SAGEConv** and different thresholds $\tau$ of graph nodes, and get the following results in 8

| **Feature** | $\tau = 0$ | $\tau = 0.01$ | $\tau = 0.05$ |
|---|---|---|---|
| profile | 73.30% | 73.76% | 66.97% |
| spacy | 77.16% | 78.19% | 75.11% |
| content | 86.88% | 88.69% | 87.33% |

Table 8: Ablation study in UPFD-Root

*Side note: We don't experiment on bert since we experienced a memory bottleneck*

### 4.3 Analysis

From the results we can draw some conclusions as follow:

1. BERT seems to be a better choice for text-based fake news detection compared to traditional models. However, the graph methods seems to leverage the probability of detection, as results increased by a large margin (5 to 10%).
2. Most of the time, **GATConv** seems to be the best choice when it comes to graph-based fake news detection.
3. **content** feature can achieve such high accuracy since it uses information from both the text and the user data, resulting in a more complex and meaningful input.
4. GossipCop and PolitiFact doesn't span every scenerio in real life, therefore it is suggested for users to finetune the model when it comees to new data.
5. A small amount connection between authors' profile can help improve the detection result, however, too much connection can harm the model, as shown in table 8.
6. Results from 8,5,4 shows that retweet helps detecting fake news by a small margin, therefore ultilizing UPFD seems to be a better choice compared to UPFD-Root.

## 5 Conclusion

This capstone project explored the application of machine learning techniques for fake news detection within the context of Web Mining. Utilizing the FakeNewsNet and UPFD datasets, we investigated various algorithms and approaches, ultimately achieving promising results in accurately identifying fake news articles.
At the same time, the availability of labeled data remains a challenge for fake news detection. Also, with the growth of Large Language Models (LLMs) and Generative AIs, the textual context tends to be less and less relevant since low quality news can easily be tailored, which is not shown in the datasets we used, dating back to 2017 and 2020.
Overall, this project demonstrated the potential of various approaches for fake news detection. By addressing the limitations and exploring future research directions, we can contribute to the development of more robust and reliable solutions for tackling this critical challenge in the digital age. We would also like to express our gratitude to PhD. Nguyen Kiem Hieu, whose invaluable guidance and insightful mentorship led us through this course.

## 6 Group members & Contribution

| Member | Student ID | Contribution |
|---|---|---|
| Nguyen Ngoc Khanh | 20204915 | Content-based fake news detection, UPFD-Root kNN |
| Tran Ngoc Khanh | 20200326 | Source-based fake news detection, UPFD-Root GNN |
| Tran Quoc Khanh | 20204881 | FakeNewsNet data collection and crawling, problem statement |
| Tran Cat Khanh | 20204916 | UPFD-Root, setup |
| Ho Minh Khoi | 20204917 | Graph-based fake news detection, experiment, result and analysis |

## References

[1] Slides, cs224w, fall 2023. http://web.stanford.edu/class/cs224w/.

[2] Leo Breiman. *Machine Learning*, 45(1):5–32, 2001.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[4] Yingtong Dou, Kai Shu, Congying Xia, Philip S. Yu, and Lichao Sun. User preference-aware fake news detection. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021.

[5] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[6] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[8] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[9] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '14. ACM, August 2014.

[10] Aditya Rayaprolu. How much data is created every day in 2023?, Jul 2023.

[11] Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. Fakenewsnet: A data repository with news content, social context and dynamic information for studying fake news on social media. *arXiv preprint arXiv:1809.01286*, 2018.

[12] Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36, 2017.

[13] Kai Shu, Suhang Wang, and Huan Liu. Exploiting tri-relationship for fake news detection. *arXiv preprint arXiv:1712.07709*, 2017.

[14] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[15] Han Xiao. bert-as-service. https://github.com/hanxiao/bert-as-service, 2018.