NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

CZ4032 Data Analytics and Mining

Report for Project 1

2022-23 Semester 1

A Study on Fake News Detection using Graph Neural Networks

Group Members:

Karanam Akshit (U2020311E)

Agarwal Pratham (U2023384F)

George Rahul (U2023835C)

Louis Wirja (U2023825E)

Chopra Dhruv (U2023974A)

Abstract

Misinformation and the spread of fake news have caused adverse effects on individuals, communities, and society in recent years, making the problem of fake news detection very relevant in current times. There are several existing solutions to this classification problem including traditional Machine Learning and Deep Learning models which are trained purely on intrinsic features extracted from the news article. However, none of the above methods utilise extrinsic features like the user propagation graph of the article to determine its validity. Therefore, the aim of this paper is to discuss in-depth our innovative Graph Neural Network (GNN) approach and compare its performance with conventional models, to investigate whether a combination of intrinsic and extrinsic features would lead to an improvement in classification accuracy. This model builds upon the model architecture from the UPFD paper, and the Gossipcop and Politifact datasets present in UPFD are used for training and benchmarking our model against a Support Vector Machine, a Logistic Regressor, and a Deep Neural Network. The experiments conducted indicate that the performance of the GNN outperformed the other models for the classifying task across the two datasets. The layers in the GNN aggregates neighbourhood information, allowing it to better learn and fit the datasets as compared to other methods. The results of this study promise a more accurate classification model than the original UPFD method. We hope that our study offers insights into the potential of GNNs for classification tasks, and can serve as a baseline for future research.

1. Introduction

1.1 Fake news

With the rise of social media platforms such as Twitter and Facebook, more users tend to seek news from these social networks as opposed to traditional media such as newspapers or television. For example, a recent survey conducted by Pew Research (Matsa, Grant, 2021) in 2021 found that about 31% of adults in the United States (U.S.) consume news from Facebook regularly, while about 13% from Twitter. This new consumption has caused an increase in the propagation of fake news - incorrect and misleading content disguised as news articles designed to influence consumers (Shu et al, 2017).

Many countries have adopted new laws to combat the rising spread of misinformation. For example, in Singapore, the Parliament passed a new law named *Protection from Online Falsehoods And Manipulation Act (POFMA)*: if there is an immediate threat to Singapore's national interest due to the spread of misinformation, social media platforms have to notify every user about it as directed by the government. Additionally, fact-checking manually is a laborious task and involves domain experts, which may be ineffective due to the sheer volume.

Various computational and deep learning methods have shown encouraging results in identifying fake news. Some examples include SAF (Zhou et al, 2020) and FakeBert (Kaliyar et al, 2021) used Convolutional Neural Networks (CNN) and BERT respectively to encode news textual information. GCNFN (Monti et al, 2019), GCN-CL (Han et al, 2020), UPFD (Dou et al, 2021) employed Graph Convolutional Networks (GCN) by encoding news propagation on social media. In this paper, we aim to improvise the model architecture employed in UPFD and also compare its performance with traditional machine learning techniques such as Support Vector Machine (SVM) and Logistic Regression.

1.2 UPFD Dataset

The UPFD Dataset is a publicly available dataset on PyTorch Geometric. The UPFD Dataset has been derived from the FakeNewsNet (Shu et al., 2020) dataset, which tackled news coming from two fact-checking websites - Gossipcop and Politifact. Using the FakeNewsNet dataset, for each news in the dataset, a Twitter propagation graph was created, where the root node represented the news, while the leaf node represents the user that has retweeted the original news article. Two user nodes have an edge if one user retweeted a tweet from the other user. The root node feature vector consists of the encoded text from the article, while that of the user consists of encoded text from his past 200 tweets and his user profile information. There are 2 types of node feature vectors in the dataset - ‘content’ and ‘bert’.

For the ‘content’ feature vector, each node has its text encoded with the Spacy Word2Vec model into a 300-dimensional vector. A 10-dimensional user profile vector which contains user information (retrieved from Twitter) is concatenated with the spacy vector, to create a 310-dimensional ‘content’ vector. For the ‘bert’ feature vector, each node has its text encoded with BERT into a 768-dimensional vector. In this paper, the ‘bert’ feature vector was used to train the models.

In-depth information on how the dataset was created can be found in the UPFD paper. The dataset statistics are shown in Table 1.1 below.

Data	**No. Graphs**	**No. Fake News**	**No. Nodes**	**No. Edges**	**Avg Nodes Per Graph**
Politifact	314	157	41,054	40,070	131
Gossipcop	5454	2,732	314,263	308,798	58

Table 1.1 UPFD Dataset Statistics

1.3 Generating Twitter Propagation Graph for Real World Data

If there is a need to use the model for a real-world news article, then the news article URL is required. Utilising the Twitter developer API, tweets that have mentioned this news article/news URL and their retweet information can be retrieved to build a news propagation graph. The graph is built based on the two following assumptions :

1. For any account v_i , if v_i retweets the same news later than at least one following accounts in $\{v_1, . . . , v_n\}$, the news spreads from the account that retweeted the latest to account v_i are estimated. The latest tweets are presented first in the Twitter timeline of the app, they therefore have higher probabilities to be retweeted.

2. The news spreads from the accounts with the greatest followers are conservatively estimated if account v_i does not follow any accounts in the retweet sequences, including the source account, as tweets from accounts with more followers are more likely to be seen and retweeted by other users.

2. Methodology

2.1 Graph Convolutional Networks (GCN)

2.1.1 Model Architecture

Deep learning techniques have impacted various fields in the past decade, especially in computer vision, speech analysis, and natural language (LeCun et al, 2015). Traditional Convolutional Networks is an example of such deep learning technique, based on key assumptions of grid structures and euclidean data. In this trained model, a variant of the GCN known as GraphSage is used instead of the traditional convolutional network. As graphs are of a non-euclidean nature, the classic convolution action on grids is replaced with a local permutation-invariant aggregation performed on the neighbours of a vertex.

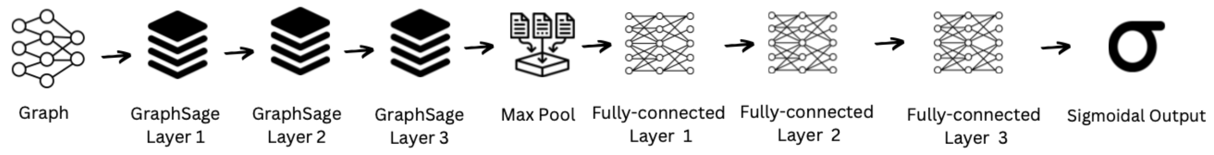


Figure 2.1.1 Model Architecture For Gossipcop Dataset

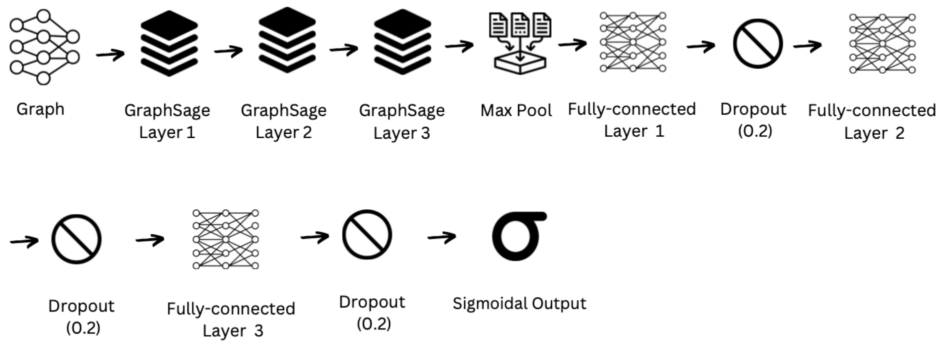


Figure 2.1.2 Model Architecture For Politifact Dataset

Figure 2.1.1 and Figure 2.1.2 above describe the architecture used for training the network. Firstly, the graph is fed through three GraphSage layers and a max pool layer is used to convert the output from the GraphSage layers to a vector, which could then be the input to the fully-connected layers. The model architecture remains largely the same for both the Politifact and Gossipcop datasets, except that for the former, dropouts with a probability of 0.2 were used for the fully-connected layers.

2.1.2 Experimental Setup

The model was implemented on PyTorch and the Graph CNN layers were implemented with PyTorch-Geometric. A batch-size of 256, binary-cross entropy loss, Adam optimizer were used to train the network. Tuning of hyperparameters was conducted and its results will be discussed below in Section 3.

2.2 Traditional Methods

Given that the UPFD dataset is the form of graphs, and that the size of each graph varies, it is not possible for many traditional methods to make full use of the graph data.

As such, only the root node, which contains the encoded (BERT-encoded) information of the news article, is used to train the following traditional methods - Support Vector Machine, Logistic Regression and a Deep Neural Network. The experiments are conducted on both Politifact and Gossipcop datasets.

2.2.1 Support Vector Machine (SVM)

SVM is a supervised learning algorithm and is widely used for classification, outlier detection and regression. Here, we use SVM for classification to get labels for ‘true’ or ‘false’ news. The algorithm is implemented using the scikit-learn library.

2.2.2 Logistic Regression

Logistic Regression is a supervised learning algorithm and is widely used for classification problems. The algorithm is implemented using the scikit-learn library.

2.2.3 Deep Neural Network

For both the Politifact and Gossipcop datasets, there are 4 fully-connected layers and a sigmoidal output layer. The neural network was implemented on Tensorflow. With the help of KerasTuner, the optimal number of units in each hidden layer and the respective learning rates were tuned. For the Politifact dataset, two types of splitting were implemented: 20:10:70, which was the original split made in the dataset (in PyTorch Geometric) and a custom split of 70:10:20. Only the original split was used for Gossipcop dataset. Table 2.2.3.1 below summarises the details of the model architecture used.

Dataset	**No. Units in 1st Layer**	**No. Units in 2nd Layer**	**No. Units in 3rd Layer**	**No. Units in 4th Layer**	**Learning Rate**
Gossipcop	140	136	204	116	0.00391
Politifact (20:10:70)	176	172	176	44	0.00943
Politifact (70:10:20)	188	148	112	112	0.0158

Table 2.2.3.1 Summary of model architectures used to train

For both datasets, binary-cross entropy loss and Adam optimizer were used in the training. Early stopping with a patience of 3 was used on the validation loss and a dropout probability of 0.5 was used on each hidden layer to reduce the chances of overfitting. Rectified Linear Unit (ReLU) was used as a non-linear activation function throughout the network.

3. Results and Discussion

3.1 Results

Table 3.1 below shows the performance of fake-news detection by the methods as described in Section 2. The highest measures are highlighted in red.

Dataset	**GCN**		**SVM**		**Logistic Regression**		**Deep Neural Network**	
	Acc	**F1**	**Acc**	**F1**	**Acc**	**F1**	**Acc**	**F1**
Gossipcop (20:10:70)	**0.99**	**0.99**	0.73	0.73	0.73	0.73	0.73	0.66
Politifact (20:10:70)	**0.85**	**0.85**	0.83	0.83	**0.85**	**0.85**	0.78	0.66
Politifact (Custom 70:10:20)	**0.93**	**0.93**	0.83	0.83	0.84	0.84	0.74	0.65

Table 3.1 Performance of models

3.2 Comparison of Performance

As per Figure 3.1, the GCN performed better at the classification task as compared to the other methods across all the considered fake news datasets.

For the Gossipcop dataset, the GCN performed with an accuracy & F1 score of 0.99, surpassing the SOTA benchmarks (Accuracy 0.98) for the dataset. We observe a clear divide between the accuracies of the GCN and other classification methods considered, with the next best accuracy achieved by these classification methods at 0.73.

For the Politifact dataset, it was discovered that the default 20:10:70 train-validation-test split by pytorch geometric left only 62 graphs in the training set. This drastically affected the performance of the GCN, with an accuracy & F1 of 0.85, making its performance comparable to other classification methods like Logistic Regression (Accuracy & F1 = 0.85) & SVM (Accuracy & F1 = 0.83). Hence, from the table above, we can conclude that on smaller train datasets, a better performance can be achieved by using simpler & more explainable methods like Logistic Regression & SVMs.

However, the GCN outperformed other methods by a margin of 0.09 in test accuracy when it was allowed to train on data with a 70:10:20 train-validation-test split, achieving an accuracy of 0.93 on test data. This demonstrates the applicability of Graph Neural Networks on larger training sets. Moreover, the model outperforms the SOTA accuracy of 0.85 for the UPFD politifact dataset.

3.3 Strengths of GNNs over traditional methods

The GCN model utilised GraphSage layers with mean aggregation, aiding it in aggregating neighbourhood information through message passing, which allowed the model to learn richer features in the considered datasets.

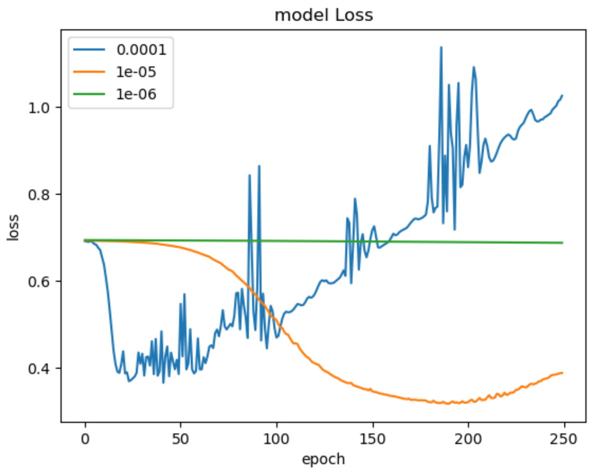
It is observed, however, that the GNN model required more training data to surpass the performance of other classification methods. For smaller datasets, traditional methods like Logistic Regression gain an edge over more advanced techniques due to their high explainability without a compromise on the performance. This is indeed true for Deep Neural Networks in general, and has been demonstrated before by other papers (Wang et al, 2018). GNN models are immune to adversarial attacks compared to other traditional methods.

3.4 Effect of hyperparameters

The values of these hyperparameters determine the accuracy of the model. As such, an optimal choice of these parameters would produce the greatest accuracy, while a sub-optimal choice may result in poorer fits. Some of the important hyperparameters to tune include: number of neurons per layer, learning rate and momentum of the Adam optimizer.

For the number of neurons in each layer, more is not always better. Increasing the size of the neurons will increase the accuracy as the model is able to sieve out more patterns, up to a certain point. Beyond that, the model will start to overfit on the training data, which in turn affects test accuracies.

The learning rate is used in adjusting the model's parameters at each epoch. If the learning rate is too small, the changes to the parameters are small as well, which would signify that more time is needed to reach the minimum point, hence more time is required to train the model. Time is an important issue as computational resources are usually limited. If the learning rate is too large, the local minima could be skipped and lead to fluctuating losses. Hence, it is necessary to find the right balance, where the learning rate is not too low or too high. Figure 3.4.2 below shows the effect of learning rate on validation loss. We can see the optimal learning rate appears to be in the range of 10^{-5}, whereas the learning rate 10^{-3} and above are yielding sub-optimal results. We can also see that for the learning rate 10^{-6}, the validation loss does not seem to change and stays constant, indicating the learning rate is very small and is unable to move forward.



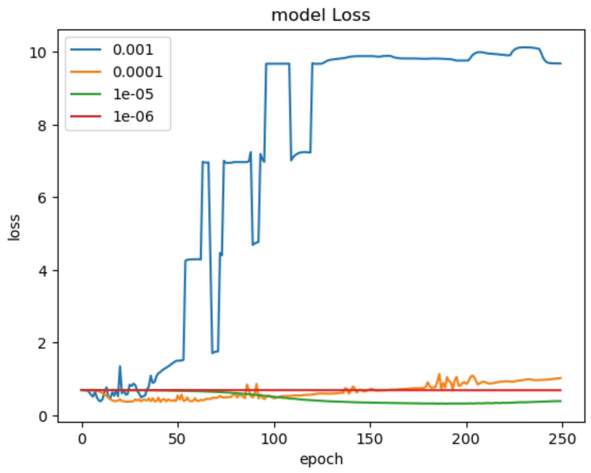


Table 3.4.1 Effect of different learning rates on validation loss(Left graph excludes 0.001 learning rate for a better scaled graph)

Momentum helps to reduce the chances of falling into a local minima by resisting rapid changes to parameter values. A good momentum value would theoretically help in reaching the global minima. Figure 3.4.2 below shows the effect of momentum of Adam optimizer on validation loss.

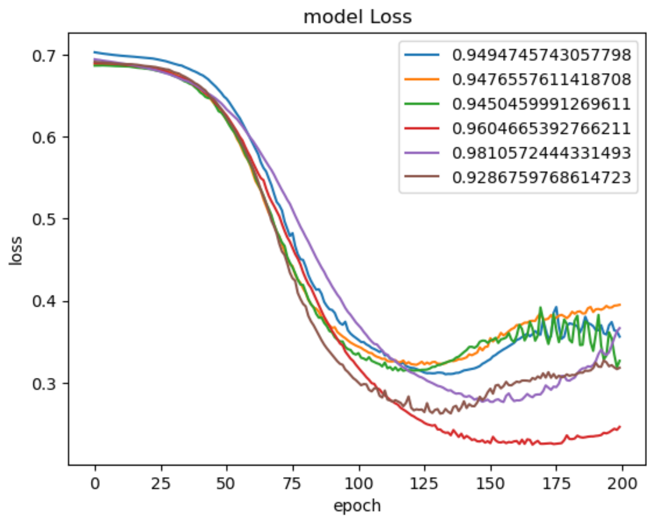


Table 3.4.2 Effect of different beta/momentum of Adam Optimizer on validation loss

The Optuna PyTorch Tuner (Akiba, et al, 2019) library was used for tuning the model's hyperparameters. Gaussian Tuning using TPE (Tree-structured Parzen Estimator) sampler was performed with Hyperband Pruner. The tuner performs an informed search in the direction of minimising the validation loss aided by the pruner. Figure 3.4.3 below shows the hyperparameters and their respective search space. Figure 3.4.4 shows the best parameter obtained after the tuning.

Hyper Parameter	**Low**	**High**	**Step**	**Sampling Method**
SAGEConv Layer Neurons	256	512	128	Uniform
Dense Layer Neuron	64	512	64	Uniform
Learning rate	$1e^{-6}$	$1e^{-3}$	NA	Log Uniform
Momentum(Adam)	$1-1e^{-1}$	$1-1e^{-3}$	NA	Log Uniform

Table 3.4.3 Search Space of all Hyperparameters (NA - default value of Optuna tuner)

Model Archetures	**Neurons SAGEConv Layer 1**	**Neurons SAGEConv Layer 2**	**Neurons SAGEConv Layer 3**	**Neurons Dense Layer 1**	**Neurons Dense Layer 2**	**Neurons Dense Layer 3**	**Learning rate**	**Momentum (Adam)**
Gossipcop Model	384	256	512	64	192	NA	1.714 $e^{-0.5}$	0.93084
Politifact Model (Custom 70:10:20)	512	512	256	64	512	64	1.433 e^{-05}	0.92867
Politifact Model (Custom 20:10:70)	512	512	256	64	64	NA	5.531 e^{-06}	0.95914

Table 3.4.4 Results after Tuning (NA symbolises model does not have that parameter)

3.5 Node feature vector and its effect

As mentioned in Section 1, the UPFD dataset has 2 types of node features - ‘content’ and ‘bert’. During experimentation on the politifact dataset, it was observed that using ‘content’ as the node feature gave a better overall accuracy, as compared to ‘bert’.

The following explanations are hypothesised to be the reasons for the results obtained. Firstly, it might be that for the genre of political news, the user features play a more significant role, and hence excluding the 10-dimensional user feature vector found in the ‘content’ feature might have brought about the differences in test accuracies. Secondly, the 768-dimensional ‘bert’ feature is of a higher dimension than that of the 310-dimension ‘content’ feature. This means that the network has to optimise more parameters and the model could be more complex to train.

Further analysis and experimentation is required to understand exactly why this difference in accuracy arises.

4. Conclusion

In this paper, we have compared the performances of various machine learning models on the Politifact (UPFD-Pol) & Gossipcop (UPFD-Gos) Fake News Datasets, with a special focus on the utility of Graph Neural Network based Machine Learning methods in this domain. We have explored the structure of the graph datasets created using twitter propagation graphs, and through the performance metrics, commented on the usefulness of the information about how the news articles spread over the content of these articles. Finally, we have analysed the model architecture of the Graph Convolution Network and have discussed the tuning of the hyperparameters using a Gaussian Hyperparameter tuner to arrive at a classification model which outperforms State of the Art approaches in both the UPFD-Pol & UPFD-Gos datasets. Along the way, we have highlighted some pros and cons of utilising a Graph Neural Network in a fake news classification domain.

5. References

1. Matsa, Grant. 2021. News Consumption Across Social Media. Pew Research Centre.

2. Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. ACM SIGKDD explorations newsletter, 19(1), 22-36.

3. Zhou, X., & Zafarani, R. (2020). A survey of fake news: Fundamental theories, detection methods, and opportunities. ACM Computing Surveys (CSUR), 53(5), 1-40.

4. Zhou, X., Wu, J., & Zafarani, R. (2020, May). SAFE: Similarity-Aware Multi-modal Fake News Detection. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 354-367). Springer, Cham.

5. Kaliyar, R.K., Goswami, A. & Narang, P. FakeBERT: Fake news detection in social media with a BERT-based deep learning approach. Multimed Tools Appl 80, 11765–11788 (2021). https://doi.org/10.1007/s11042-020-10183-2

6. Monti, F., Frasca, F., Eynard, D., Mannion, D., & Bronstein, M. M. (2019). Fake news detection on social media using geometric deep learning. arXiv preprint arXiv:1902.06673.

7. Han, Y., Karunasekera, S., & Leckie, C. (2020). Graph neural networks with continual learning for fake news detection from social media. arXiv preprint arXiv:2007.03316.

8. Dou, Y., Shu, K., Xia, C., Yu, P. S., & Sun, L. (2021, July). User preference-aware fake news detection. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 2051-2055).

9. Shu, K., Zheng, G., Li, Y., Mukherjee, S., Awadallah, A. H., Ruston, S., & Liu, H. (2020). Leveraging multi-source weak social supervision for early detection of fake news.

10. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

11. Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. Advances in neural information processing systems, 30.

12. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., & Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. Advances in neural information processing systems, 31.

13. Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019, July). Optuna: A next-generation hyperparameter optimization framework. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 2623-2631).

14. Wang, T., Huan, J., & Li, B. (2018, November). Data dropout: Optimizing training data for convolutional neural networks. In 2018 IEEE 30th international conference on tools with artificial intelligence (ICTAI) (pp. 39-46). IEEE.

15. Protection from Online Falsehoods and Manipulation Act 2019 (2019). Available From: https://sso.agc.gov.sg/Act/POFMA2019?TransactionDate=20191001235959