APMTH 220 Geometric Methods for Machine Learning **Spring 2024**

User Preference Driven Fake News Detection

Name: Mayesha Soshi *Submission Date: May 3, 2024*

Abstract

This project aims to develop effective fake news detection models using Graph Neural Networks (GNNs) and the User Preference-aware Fake News Detection (UPFD) framework. The proliferation of fake news on social media platforms has emerged as a significant global challenge, undermining public trust and manipulating opinions. Traditional content-based detection methods often fall short in capturing the nuanced and evolving nature of fake news. Therefore, this project leverages the UPFD framework, which jointly models user preferences and social context, to build three GNN-based models: a Graph Convolutional Network (GCN), a Bi-directional GCN (Bi-GCN), and a Graph Attention Network (GAN). The models are evaluated on two datasets, PolitiFact and GossipCop, using metrics such as accuracy and F1 score. The results demonstrate the effectiveness of the GAN model, which outperforms the GCN and Bi-GCN models on both datasets.

1 Introduction

In the era of social media, the rapid dissemination of fake news has emerged as a significant global challenge. Fake news, defined as intentionally and verifiably false information published in the guise of legitimate news, can manipulate public opinion, erode trust in media, and even influence high-stakes events such as elections. The proliferation of fake news is exacerbated by the ease of content creating and sharing in social media platforms, where false information can quickly go viral and reach a wide audience. Detecting fake news is a complex task, as it requires understanding not only the content of the news itself, but also the social context and engagement patterns surrounding it. Traditional content-based detection methods often fall short in capturing the nuanced and evolving nature of fake news. Therefore, there is an urgent need for advanced techniques that can effectively identify fake news by leveraging the rich relational information available in social networks. Graph Neural Networks (GNNs) have emerged as a promising approach to tackle this challenge as they can model complex interactions between users, news articles, and the social context to uncover patterns indicative of fake news propagation.

Previous work on fake news detection primarily focused on extracting features from text contexts, user profiles, and propagation structures to learn a classifier from labeled data. Text context based methods typically rely on linguistic characteristics to identify deceptive cues or writing styles. However, they fail to be comprehensive as they can be undermined by sophisticated fake news articles that do not immediately seem false. In contrast, social context features encompass user demographics (such as age, gender, education, etc.), social network structure, and user interactions to classify fake news. Propagation-based approaches, on the other hand, analyze how news spreads over time. The fundamental premise behind propagation-based approaches is the assumption that a news story correlates strongly with the reliability of associated social media posts. Propagation-based methods construct homogeneous and heterogeneous credibility networks for the propagation process. Homogeneous credibility networks comprise a singular entity type, like posts or events while heterogeneous credibility networks encompass various entity types, such as posts, sub-events, and events.

Although these methods represent significant progress, there remains a demand for more sophisticated techniques to develop even more precise fake news detection algorithms. Recent research has explored the application of GNNs for fake news detection, leveraging the ability of GNNs to model

complex relationships and propagation patterns in social networks. Monti et al. (2019) proposed the GCNFN model, which employs Graph Convolutional Networks (GCNs) to learn representations of users and news articles based on their engagement patterns and propagation paths in a social graph. They demonstrate the effectiveness of this approach in capturing the structural information for fake news detection. Similarly, Han et al. (2020) introduced a GNN-based model with continual learning to adapt to the evolving nature of fake news. Their model incorporates temporal information and allows for incremental updates to detect newly emerged fake news. Bian et al. (2020) proposed a Bi-directional GCN (Bi-GCN) model that considers both top-down and bottom-up propagation of news in a social graph, capturing the bidirectional influences between news articles and users. While a great advancement, these methods only focus on modeling news content and its user exogenous context. The User Preference-aware Fake News Detection (UPFD) framework proposed by Dou et al. (2021) represents a significant advancement over previous methods for fake news detection as it leverages both the endogenous user preferences and the exogenous social context to better identify fake news. To model the user's endogenous preference, Dou et al. encodes news content and user historical posts using various text representation learning approaches. The dataset created by Dou et al. has been made available on PyTorch and is leveraged in this project to create fake news classification models.

2 Background and Notations

We will use the UPFD framework to build our fake news classification models. Let's take a closer look at the UPFD framework and how it was constructed. The framework consists of three main components:

1. The endogenous preference encoder, which captures user preferences by encoding news content and user historical posts using text representation learning techniques such as word2vec or BERT.

2. The exogenous context encoder, which models the social context by building a tree-structured propagation graph for each news article based on its sharing cascade on social media platforms. The news post serves as the root node, while other nodes represent users who shared the news.

3. The information fusion module, which integrates the endogenous and exogenous information by using the vector representations of news and users as node features in the propagation graph.

GNNs are then employed to learn a joint user engagement embedding. Finally, the user engagement embedding and news textual embedding are concatenated and fed into a neural classifier to predict the credibility of the news articles. Using the UPFD framework, we will construct three different models: a graph convolutional network, a Bi-directional GCN (Bi-GCN), and a graph attention network (GAN).

2.1 Graph Convolutional Networks (GCNs)

Graph Convolutional Networks (GCNs) are a class of neural networks designed to operate on graph-structured data. Given an input graph $G = (V, E)$, where V represents the set of nodes and E represents the set of edges, the goal of GCNs is to learn node representations that capture both the graph structure and node features. The key operation in GCNs is the graph convolution, which can be mathematically defined as:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

where $H^{(l)}$ represents the node features at the l-th layer, $W^{(l)}$ denotes the learnable weight matrix, $\tilde{A}$ is the adjacency matrix of the graph with added self-connections, and $\tilde{D}$ is the corresponding degree matrix. σ is an activation function, typically ReLU. Through successive graph convolutional layers, GCNs can learn hierarchical representations of the input graph.

2.2 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) are a class of generative models that consist of two neural networks, a generator G and a discriminator D, trained simultaneously in a competitive setting. The generator aims to generate realistic samples from a given distribution, while the discriminator aims to distinguish between real and fake samples. The training process can be formalized as a minimax game, where the generator tries to minimize the following objective function while the discriminator tries to maximize it:

$$\min_G \max_D E_{x \sim p_{\text{data}}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

Here, $p_{\text{data}}(x)$ represents the true data distribution, $p_z(z)$ represents the prior distribution of the latent space, x represents real samples, and z represents latent variables sampled from $p_z(z)$. Through adversarial training, GANs can learn to generate samples that are indistinguishable from real data.

2.3 Bi-Directional Graph Convolutional Networks (Bi-GCNs)

Bi-directional Graph Convolutional Networks (Bi-GCNs) consists of two components: a top down GCN (TD-GCN) and a bottom up GCN (BU-GCN). TD-GCN captures the propagation pattern of rumors by forwarding information from parent nodes to child nodes along the propagation graph. It learns the causal features of rumor propagation along relationship chains from top down. BU-GCN captures the dispersion pattern of rumors by aggregating information from children nodes to parent nodes along the dispersion graph. It obtains the structural features from rumor dispersion within communities through bottom-up gathering.

3 Proposed Approach

3.1 Dataset

To implement the fake news detection model, we will use the UPFD dataset available on PyTorch. The UPFD dataset is built off of the FakeNewsNet dataset created by Shu et al. The FakeNewsNet dataset contains two comprehensive datasets, PolitiFact and GossipCop, with diverse features in news content, social context, and spatiotemporal information. The UPFD dataset statistics are included in Table 1.

Dataset	Graphs	Total Nodes	Total Edges	Avg. Nodes per Graph
Politifact	314	41,054	40,740	131
Gossipcop	5464	314,262	308,798	58

Table 1: UFPD Dataset Statistics

For a single graph in the UPFD dataset, the root node represents the source news, and leaf nodes represent Twitter users who retweeted the same root news. A user node has an edge to the news node if and only if the user retweeted the root news directly. Two user nodes have an edge if and only if one user retweeted the root news from the other user. Four different node features are encoded using different encoders. You can select the different feature encoding by updating the `feature` parameter of the UPFD dataset. The `feature` parameter can be set to one of four values: `profile`, `spacy`, `bert`, and `content`.

- `profile`: This parameter creates a 10-dimensional node feature composed of ten Twitter user profile attributes.
- `spacy`: This parameters creates a 300-dimensional node feature composed of Twitter user historical tweets encoded by the spaCy word2vec encoder.

- `bert`: This parameter creates a 768-dimensional node feature composed of Twitter user historical tweets encoded by the bert-as-service.
- `content`: This parameters creates a 310-dimensional node feature composed of a 300-dimensional "spacy" vector plus a 10-dimensional "profile" vector.

For our purposes, we will use the `content` feature parameter. Now let's take a look at the three model implementations.

3.2 GCN and GAN Model

The proposed GCN and GAN model for fake news detection consists of three main components: graph convolutions, pooling, and readout layers.

The model utilizes three Graph Convolutional (GCNConv) or Graph Attention Convolutional (GATConv) layers to learn node representations. The input node features and the edge index matrix are passed through the GATConv layers, which compute node embeddings by aggregating information from neighboring nodes. The node embeddings are updated through the GCNConv/GATConv layers, capturing the local structural information of the graph. ReLU activation is applied after each GCNConv/GATConv layer to introduce non-linearity.

After the graph convolutions, a Global Max Pooling (GMP) operation is performed on the node embeddings to obtain a graph-level representation. The GMP operation aggregates the node embeddings across the entire graph, capturing the most salient features.

The readout phase combines the graph-level representation with the raw word2vec embeddings of the news articles. First, the graph-level representation is passed through a linear transformation followed by a ReLU activation. Then, the raw word2vec embeddings of the news articles are obtained based on the root nodes of each graph in the batch. The news embeddings are transformed using a linear layer and ReLU activation. Finally, the transformed graph-level representation and the transformed news embeddings are concatenated and passed through a final linear layer to obtain the output predictions. A sigmoid activation is applied to the output to obtain probabilities for the binary classification task.

The model is trained using binary cross-entropy loss and optimized using gradient descent. The GNN architecture enables the model to capture both the local graph structure and the textual information of the news articles, leveraging the relational information in the graph and the content-based features for fake news detection.

Please refer to the Appendix for code snippets and link.

3.3 Bi-GCN Model

The proposed Bi-directional Graph Convolutional Network (Bi-GCN) model is designed to capture both top-down and bottom-up propagation patterns in the news dissemination process, as introduced in Bian et al. The model consists of two parallel branches: a top-down GCN and a bottom-up GCN, followed by a concatenation layer and a fully connected layer for final predictions.

The top-down GCN branch captures the propagation of information from the source nodes to the target nodes in the graph. It consists of two GCNConv layers. The first layer takes the input node features and the edge index matrix and applies a graph convolution operation to compute hidden representations of the nodes. The output of the first layer is passed through a ReLU activation function. The second layer takes the output of the first layer and applies another graph convolution operation to obtain the final top-down node representations.

The bottom-up GCN branch captures the propagation of information from the target nodes back to the source nodes. It has a similar structure to the top-down GCN, with two GCNConv layers and a ReLU activation function in between. The bottom-up GCN processes the input node features and the edge index matrix to compute the bottom-up node representations.

The outputs of the top-down and bottom-up GCNs are concatenated along the feature dimension using the torch.cat function. This concatenated representation captures both the top-down and bottom-up propagation patterns in the graph. The concatenated representation is then passed through a fully connected layer to obtain the final node representations.

To obtain a graph-level representation, global mean pooling is applied to the output of the fully connected layer. The global mean pool function computes the mean of the node representations across the entire graph, resulting in a single vector representation for each graph in the batch. During the forward pass, the input node features and the edge index matrix are passed through the top-down and bottom-up GCNs separately. The outputs of both GCNs are concatenated and processed by the fully connected layer. Finally, global mean pooling is applied to obtain the graph-level representations, which can be used for downstream tasks such as fake news classification.

The Bi-GCN model leverages the bidirectional propagation of information in the graph, capturing both the top-down and bottom-up patterns. By considering the information flow in both directions, the model can learn more comprehensive representations of the news dissemination process. The concatenation of the top-down and bottom-up representations allows the model to capture the interplay between the source and target nodes in the graph.

Please refer to the Appendix for code snippets and link.

4 Results

Table 2 and 3 below show the train and test accuracies of the three models on the PolitiFact and GossipCop datasets.

Model	Train Acc (%)	Test Acc (%)	F1 Score
GCN	0.5806	0.8145	0.7876
Bi-GCN	0.5806	0.7647	0.8060
GAN	0.9194	**0.8597**	0.8692

Table 2: PolitiFact Train and Test Accuracies

Model	Train Acc (%)	Test Acc (%)	F1 Score
GCN	0.8855	0.9315	0.9320
Bi-GCN	0.9020	0.9378	0.9371
GAN	0.9029	**0.9459**	0.9458

Table 3: GossipCop Train and Test Accuracies

As we can see from the table above, on the PolitiFact dataset, the GAN model achieves the highest test accuracy of 85.97% and an F1 score of 0.8692, outperforming both the GCN and Bi-GCN models. The GCN model obtains a test accuracy of 81.45% and an F1 score of 0.7876, while the Bi-GCN model has a slightly lower test accuracy of 76.47% but a higher F1 score of 0.8060. It is interesting to note that both the GCN and Bi-GCN models have the same training accuracy of 58.06%, suggesting that they have similar learning capacities on the training data. However, the GAN model demonstrates better generalization ability on the test set, as evidenced by its higher test accuracy and F1 score.

The superior performance of the GAN model on the PolitiFact dataset can be attributed to its ability to learn more discriminative and robust representations of the graph structure through the adversarial training process. The generator and discriminator networks in the GAN model work together

to capture the complex patterns and characteristics of fake news propagation. The generator aims to produce realistic graph embeddings that are difficult for the discriminator to distinguish from real ones, while the discriminator tries to correctly identify the generated embeddings. This adversarial training process encourages the model to learn more generalized and informative representations, leading to better performance on the test set.

On the GossipCop dataset, the GAN model again achieves the highest test accuracy of 94.59% and an F1 score of 0.9458. The Bi-GCN model follows closely with a test accuracy of 93.78% and an F1 score of 0.9371, while the GCN model has a slightly lower test accuracy of 93.15% and an F1 score of 0.9320. The competitive performance of the Bi-GCN model on the GossipCop dataset can be explained by its ability to capture both top-down and bottom-up propagation patterns in the graph. By considering the information flow from source nodes to target nodes (top-down) and from target nodes back to source nodes (bottom-up), the Bi-GCN model can learn more comprehensive representations of the news dissemination process. This bidirectional approach proves to be particularly effective on the GossipCop dataset, where the propagation patterns may exhibit more complex and nuanced characteristics compared to the PolitiFact dataset.

Compared to the PolitiFact dataset, all three models exhibit higher training accuracies on the GossipCop dataset. This suggests that the models are able to learn more effectively from the GossipCop dataset during training. The higher training accuracies on the GossipCop dataset can be attributed to the larger size and potentially more diverse set of examples present in this dataset. With more training samples, the models can better capture the underlying patterns and characteristics of fake news propagation, leading to improved learning performance.

Comparing the results across both datasets, it is evident that the GAN model consistently outperforms the other two models in terms of test accuracy and F1 score. The Bi-GCN model also shows competitive performance, especially on the GossipCop dataset, where it achieves comparable results to the GAN model. The GCN model, while still performing well, has slightly lower test accuracies and F1 scores compared to the other two models.

These results underscore the potential of employing advanced GNN architectures, such as GAN and Bi-GCN, for fake news detection tasks. The ability of these models to learn rich representations of the graph structure and propagation patterns contributes to their superior performance compared to the basic GCN model. However, it is important to consider the specific characteristics and size of the dataset when selecting the most appropriate model for fake news detection.

5 Discussion and Conclusions

The results demonstrate the effectiveness of GNNs in fake news detection on social media datasets. The GAN model consistently outperforms the GCN and Bi-GCN models in terms of test accuracy and F1 score on both the PolitiFact and GossipCop datasets, indicating its effectiveness in capturing the complex patterns and characteristics of fake news propagation.

The superior performance of the GAN model can be attributed to its adversarial training approach. By incorporating a discriminator network that learns to distinguish between real and fake news graphs, the GAN model is able to learn more robust and discriminative representations of the graph structure. The adversarial training process encourages the model to generate graph embeddings that are difficult for the discriminator to distinguish, thereby enhancing the model's ability to detect fake news accurately.

The Bi-GCN model also demonstrates competitive performance, particularly on the GossipCop dataset. The bidirectional propagation mechanism employed by Bi-GCN allows it to capture both top-down and bottom-up information flow in the graph, enabling it to learn more comprehensive representations of the news propagation patterns. This bidirectional approach proves to be effective in detecting fake news, as evidenced by the high test accuracy and F1 score achieved by Bi-GCN on the GossipCop dataset.

On the other hand, the GCN model, while still performing reasonably well, has lower test accuracies

and F1 scores compared to the GAN and Bi-GCN models. The GCN model relies on a simpler graph convolution operation and may not be able to capture the complex interactions and propagation patterns as effectively as the more advanced models. However, it is important to note that the GCN model still achieves good performance, indicating the potential of even basic GNN architectures in fake news detection.

The results of this study have important implications for real-world fake news detection systems. While the proposed GNN models demonstrate promising performance, deploying them in practice presents several challenges. One major challenge is dealing with data scarcity, as obtaining large amounts of labeled fake news data can be difficult and time-consuming. Additionally, fake news patterns and tactics constantly evolve, requiring the models to adapt to new forms of misinformation. Addressing these challenges is crucial for developing effective and practical fake news detection systems.

5.1 Future Directions

Future work in this area could focus on several directions to improve the models and address the limitations identified in this study. One potential avenue is to explore more advanced GNN architectures that can better capture the complex interactions and propagation patterns in fake news graphs. Another direction is to incorporate additional features, such as user metadata, temporal information, and multi-modal data (e.g., images and videos), to provide a more comprehensive representation of the fake news ecosystem. Additionally, investigating transfer learning approaches could help alleviate the issue of data scarcity by leveraging knowledge from related domains or pre-trained models.

6 References

Bian, T., Xiao, X., Xu, T., Zhao, P., Huang, W., Rong, Y., & Huang, J. (2020, April). Rumor detection on social media with bi-directional graph convolutional networks. In Proceedings of the AAAI conference on artificial intelligence (Vol. 34, No. 01, pp. 549-556).

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).

Gupta, M., Zhao, P., & Han, J. (2012, April). Evaluating event credibility on twitter. In Proceedings of the 2012 SIAM international conference on data mining (pp. 153-164). Society for Industrial and Applied Mathematics.

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.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

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.

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.

7 Appendix

```
class GCN(torch.nn.Module):
    def __init__(self, in_channels, hidden_channels, out_channels):
        super().__init__()
        # Graph Convolutions
        self.conv1 = GCNConv(in_channels, hidden_channels)
        self.conv2 = GCNConv(hidden_channels, hidden_channels)
        self.conv3 = GCNConv(hidden_channels, hidden_channels)

        # Readout
        self.lin_news = Linear(in_channels, hidden_channels)
        self.lin0 = Linear(hidden_channels, hidden_channels)
        self.lin1 = Linear(2*hidden_channels, out_channels)

    def forward(self, x, edge_index, batch):
        # Graph Convolutions
        h = self.conv1(x, edge_index).relu()
        h = self.conv2(h, edge_index).relu()
        h = self.conv3(h, edge_index).relu()

        # Pooling
        h = global_mean_pool(h, batch)

        # Readout
        h = self.lin0(h).relu()

        root = (batch[1:] - batch[:-1]).nonzero(as_tuple=False).view(-1)
        root = torch.cat([root.new_zeros(1), root + 1], dim=0)
        # root is e.g. [ 0, 14, 94, 171, 230, 302, ... ]
        news = x[root]
        news = self.lin_news(news).relu()

        out = self.lin1(torch.cat([h, news], dim=-1))
        return torch.sigmoid(out)


class BiGCN(torch.nn.Module):
    def __init__(self, in_channels, hidden_channels, out_channels):
        super(BiGCN, self).__init__()
        self.conv1_td = GCNConv(in_channels, hidden_channels)
        self.conv2_td = GCNConv(hidden_channels, out_channels)
        self.conv1_bu = GCNConv(in_channels, hidden_channels)
        self.conv2_bu = GCNConv(hidden_channels, out_channels)
        self.fc = Linear(2*out_channels, out_channels)

    def forward(self, x, edge_index, batch):
        # Top-down propagation
        x_td = self.conv1_td(x, edge_index)
        x_td = torch.relu(x_td)
        x_td = self.conv2_td(x_td, edge_index)
```

```
        # Bottom-up propagation
        x_bu = self.conv1_bu(x, edge_index)
        x_bu = torch.relu(x_bu)
        x_bu = self.conv2_bu(x_bu, edge_index)

        # Concatenate top-down and bottom-up outputs
        x_concat = torch.cat((x_td, x_bu), dim=1)
        x_out = self.fc(x_concat)

        # Apply global mean pooling
        x_out = global_mean_pool(x_out, batch)

        return x_out



class GNN(torch.nn.Module):
    def __init__(self, in_channels, hidden_channels, out_channels):
        super().__init__()

        # Graph Convolutions
        self.conv1 = GATConv(in_channels, hidden_channels)
        self.conv2 = GATConv(hidden_channels, hidden_channels)
        self.conv3 = GATConv(hidden_channels, hidden_channels)

        # Readout
        self.lin_news = Linear(in_channels, hidden_channels)
        self.lin0 = Linear(hidden_channels, hidden_channels)
        self.lin1 = Linear(2*hidden_channels, out_channels)

    def forward(self, x, edge_index, batch):
        # Graph Convolutions
        h = self.conv1(x, edge_index).relu()
        h = self.conv2(h, edge_index).relu()
        h = self.conv3(h, edge_index).relu()

        # Pooling
        h = gmp(h, batch)

        # Readout
        h = self.lin0(h).relu()

        # According to UPFD paper: Include raw word2vec embeddings of news
        # This is done per graph in the batch
        root = (batch[1:] - batch[:-1]).nonzero(as_tuple=False).view(-1)
        root = torch.cat([root.new_zeros(1), root + 1], dim=0)
        # root is e.g. [   0,   14,   94,  171,  230,  302, ... ]
        news = x[root]
        news = self.lin_news(news).relu()
```

```
        out = self.lin1(torch.cat([h, news], dim=-1))
        return torch.sigmoid(out)
```

The code can be accessed here.