Learning From Networks

Fake news detection using Graph Neural Networks

Victor Goubet - 2039343
UNIPD
Computer engineering

Nicholas Tagliapietra - 1242481
UNIPD
Mathematical Engineering

Asma Bakhtiariazad - 2050788
UNIPD
ICT

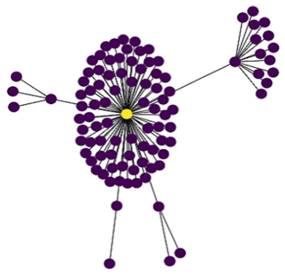
1 Abstract

Fake news are the new plague of the 21st century. With the advent of social networks and the easy and quick access to information, this disease has become more and more common. Through retweets, shares or likes, a piece of fabricated information can in a few moments gain real credibility thanks to the common "validation". Similar to the tragedy of the common, each user is selfish and does not take the time to verify the sources, preferring to believe in this information that is often incredible, revolutionary and built to make the buzz. However, for several years now, various methods have been used on social networks to address these problems by detecting and removing problematic messages from the platform as quickly as possible. Our objective is to analyse the state of the art of these methods, to implement a graph-based solution and to see if adding node-level features helps to increase the prediction score. We will take the example of Twitter and model a graph for each tweet posted. Finally, we will use the a version of the FakeNewsNet dataset [5] which is one of the reference datasets for this type of task.

2 Dataset

We will use a different version of the FakeNewsNet dataset called UPFD[7]. The advantages of this Framework is that it is available directly in Pytorch Geometric, that it doesn't need Twitter APIs (which in our case were very limiting), and also that various Node-Level Features and Embeddings have been already computed thanks to different techniques explained in: [7].

The Dataset is composed of several thousands graphs in the form of a tree where the root node is the source news and all the linked nodes corresponds to the users that tweeted the news. The edges represent the sharing history between users (not between each Tweet).



Here we recognize the news in yellow and each user who retweeted in purple. The goal is to solve a Graph Classification task where each Graph must be classified into Fake-News or Not-Fake-News. The Dataset is divided in Gossip News (Gossipcop) and Political News (Politifact), and when loading the Dataset we can choose freely which part to use (Gossipcop) and also what node-level features to adopt (User Profile Info, Embedded History of the user's tweets). The node-features had been extracted and pre-computed in [7] thanks to the Spacy word2vec encoder or BERT. Finally, the Dataset contains 5464 graphs corresponding to real news and 2732 fake news, each graph being on average composed of 58 nodes (A).

3 Intended experiments

The goal is simple: To study how the Graph-Level and Node-Level features contributes to the final result. We will extract new features and validate/reject all of them by testing them on different models.

As seen before, the UPFD Framework provides already various Node-Level features:

- **Profile feature**: 10 node feature of user profile attributes.
- **Spacy features:** 300 node feature of user historical tweets [1]
- **BERT feature**: 768 node feature of user historical tweets [6]

Some of the new features that can be extracted are:

Node-level features

Possible Node-Level features that will be tested are: [**Degree, Closeness centrality, Betweenness centrality, etc.**]. Once these new features are computed, we can merge them to the existing feature vector. Different combinations and fusion of features will be made in order to maximize the results, e.g. by comparing the concatenation or the multiplication of some of them.

Graph-level features

For each graph we can calculate: [**Average degree, Degree of the root node, Number of nodes, Graphlet count, etc.**]. Features fusion could be done just before the linear layer of a GNN (e.g. after a mean-max pooling) but, as before, different strategies will be tested.

Finally, we will build various models (GCN[4], GraphSAGE [3], GAT [8], et similia.) each one based on UPFD (B), test with and without different features, and then compute the accuracy of these models. In such a way we will be able to identify the relevant features and the best models. Everything will be developed with Pytorch Geometric [2] and NetworkX.

References

[1] Tomas Mikolov Kai Chen Greg Corrado and Jeffrey Dean. “Efficient estimation of word representations in vector space”. In: (2013).

[2] Matthias Fey and Jan E. Lenssen. “Fast Graph Representation Learning with PyTorch Geometric”. In: (2019).

[3] William L. Hamilton, Rex Ying, and Jure Leskovec. “Inductive Representation Learning on Large Graphs”. In: (2017).

[4] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: (2016). arXiv: 1609.02907.

[5] Kai S. Deepak M. Suhang W. Dongwon L. “FakeNewsNet: A Data Repository with News Content, Social Context and Spatialtemporal Information for Studying Fake News on Social Media”. In: (2018).

[6] Jacob Devlin Ming-Wei Chang Kenton Lee and Kristina Toutanova. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: (2018).

[7] Yingtong D. Kai S. Congying X. Philip S. Lichao S. “User Preference-aware Fake News Detection”. In: (2021).

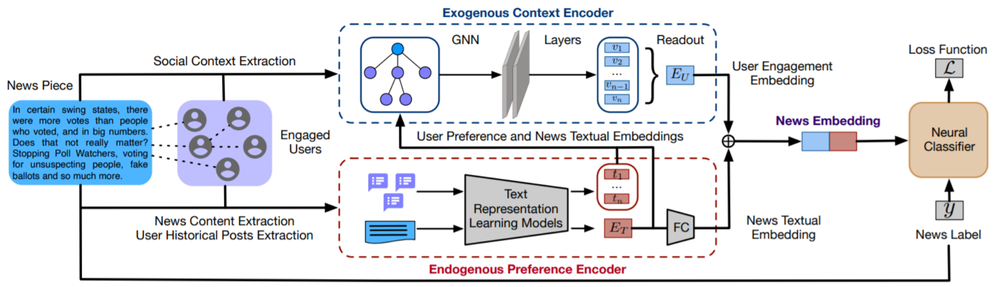
[8] Petar Veličković et al. “Graph Attention Networks”. In: (2018).

A Dataset statistics

Dataset	**Graphs (Fake)**	**Total Nodes**	**Total Edges**	**Avg. Nodes/Graph**
Politifact (POL)	314 (157)	41.054	40.740	131
Gossipcop (GOS)	5464 (2732)	314.262	308.798	58

Table 1: Statistics of the two datasets

B Architecture of Fake News Detection Network in UPFD


Exogenous Context Encoder
GNN
Layers
Readout
v_1
v_2
v_{n-1}
v_n
E_U
Social Context Extraction
News Piece
In certain swing states, there were more votes than people who voted, and in big numbers. Does that not really matter? Stopping Poll Watchers, voting for unsuspecting people, fake ballots and so much more.
Engaged Users
User Engagement Embedding
Loss Function
$\mathcal{L}$
News Embedding
Neural Classifier
User Preference and News Textual Embeddings
Text Representation Learning Models
t_1
t_n
E_T
FC
News Content Extraction
User Historical Posts Extraction
News Textual Embedding
y
News Label
Endogenous Preference Encoder