On The Pursuit of Fake News : From Graph Convolutional Networks to Time Series

Zeynep Pehlivan
Institut national de l'audiovisuel, France
zpehlivan@ina.fr

ABSTRACT

This paper presents the methods proposed by team INAFake team for MediaEval 2020 FakeNews: Corona virus and 5G conspiracy. We concentrate our work on the sub-task of structure-based fake news detection. Our aim is to test existing methods by leaning on temporal features of networks without taking any textual features into account. We applied two well known supervised graph classification approaches, graph convolutional layers (GCN) and Deep Graph Convolutional Neural Network (DGCNN). We also present the problem as a multivariate time series classification problem and tested multivariate long short term memory fully convolutional network method.

1 INTRODUCTION

Social media, which provides instant textual and visual information exchange, plays an important role in information propagation but plays also a crucial role for the propagation of fake information. One study [1] estimates that 42 percent of visits to fake news websites came through social media. Specially, when fake news distort real-world information by tweaking or mixing it with the true information, it spreads faster on social media [10].

The aim of Fake News task [9] is to detect misinformation spreaders by analysing tweets related to Coronavirus and 5G conspiracy - the idea that the COVID-19 outbreak is somehow connected to the introduction of the 5G wireless technology. The challenge of this task is not only to detect the fake news but also to make the distinction between fake news related to Corona virus-5G and other fake news subjects. This work addresses the issues related to sub-task of structure-based fake news detection, thus it does not take the tweets content into account.

2 RELATED WORK

Fake news detection focuses on using news contents and social contexts. For social context based approaches, the features mainly include user-based, post-based and network-based. For this challenge, we will focus on network based features. Two graph learning problems have been well studied: node classification and graph classification. Node classification is to predict the class label of nodes in a graph, while graph classification aims to predict the class label of graphs, for which various graph kernels and deep learning approaches have been designed.

We first apply two different graph classification algorithms to this challenge's dataset. First one is based on graph convolutional classification model, GCN, proposed in [8] by using the graph convolutional layers from [7]. However, [7] is developed for node classification, it can be extracted to use with the graph classification by implementing a global pooling layer as a last layer that performs some form of pooling operation [1].

The second approach is the Deep Graph Convolutional Neural Network (DGCNN) [13] algorithm. It uses the graph convolutional layers from [7] and proposes "SortPooling" which sorts nodes according the concatenation of the node embeddings of all layers as the continuous equivalent of node coloring algorithms. Then, such "colors" define a lexicographic ordering of nodes across graphs. The top ordered nodes are then selected and fed (as a sequence) to a one-dimensional convolutional layer that computes the aggregated graph encoding. roles within the graph [2].

As studied in [10], fake news spread significantly faster and deeper than the truth. Thus, we would like to put the temporal dimension into account for this challenge and problem falls into time series classification category. Instead of creating univariate time series from tweets published/retweet dates, we create multivariate time series (MLTS) by using graph features changing over time. Recently, most approaches to MLTS have used neural networks, and in particular convolutional neural networks [6, 12]. We use MLTSM-FCN [6] which is a combination of long short term memory (LSTM) [5] and one-dimensional fully convolutional networks (FCN) [11] joined by a concatenation layer, followed by a shared dense layer for predictions. In [6], authors propose two versions, one with attention layer (MALTSM-FCN), one without attention layer. We choose to use the version with attention layer.

3 APPROACHES

In this section, we are going to give details of our implementations for GCN, DGCNN and MALTSM-FCN.

3.1 GCN

Our deep learning model is represented in Figure 1. The input is the graph represented by its adjacency and node features matrices. The first three layers are Graph Convolutional as in [7] with each layer having 128 units with relu activations and orthogonal kernel initializer. The next layer is a mean pooling layer where the learned node representation are summarized to create a graph representation. The graph representation is input to three fully connected layers with 128, 32 and 16 units respectively with relu activations and orthogonal kernel initializer. The model is trained using a batch size of 128.

MediaEval'20, December 14-15 2020, Online

[1] https://github.com/tkipf/gcn/issues/4

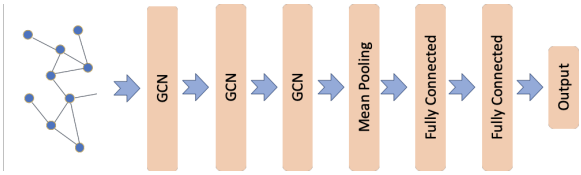

Figure 1: GCN architecture from [3]

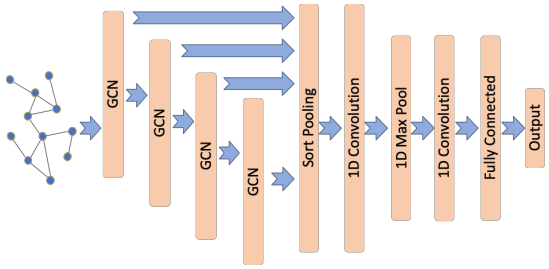

Figure 2: DGCNN architecture from [3]

3.2 DGCNN

The model is represented in Figure 2. The model's input is also the graph represented by its adjacency and node features matrices. The first four layers are Graph Convolutional layers, each have 128, 128, 128, 3 units and tanh activations. These layers are followed by a one dimensional convolutional layer, Conv1D, followed by a max pooling, MaxPool1D, layer. Next is a second Conv1D layer that is followed by two Dense layers, first one with relu activation and followed by droput layer (0.2) and second one with softmax activation for classification.

For GCN and DGCNN, categorical cross-entropy loss is used to train the neural network. The models are trained using a batch size of 128 and 256 respectively, Adam optimizer with initial learning rate 0.001 and decay 0.01, with dropout (0.2). We also reduced the learning rate by a factor 1/10 and applied early stopping. StellarGraph [3] and networkx [4] packages are used for the implementation.

3.3 MALSTM-FCN

This model is represented in Figure 3. The model's input is the time series generated by using graph features explained below. This model is implemented by using source code of [6] [2]. For the MALSTM-FCN network, the optimal number of LSTM hidden states for each dataset was found via grid search over 8, 16, 32. The FCN block consists of three blocks of 128-256-128 filters. The models are trained using a batch size of 128. He uniform initializer is used for the convolution kernels. The activcation function is set to relu.

3.4 Input generation

For GCN and DGCNN, the same input is used. As explained in the challenge, the provided retweet graphs contain sub-graphs of the Twitters follower graph and as suggested by the organizers, since each sub-graph must contain the trajectories of the real world

[2]https://github.com/titu1994/MLSTM-FCN

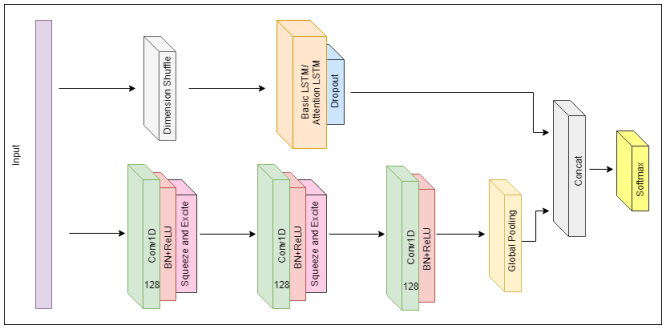

Figure 3: MALSTM-FCN architecture from [6]

spreading we pre-processed the provided graphs and discarded all edges that point against the time. Then, for each node in each graph, following features are calculated by using networkx package : degree centrality, closeness centrality, betweenness centrality,load centrality, harmonic centrality, number of cliques, clustering coefficient, square clustering coefficient and average neighbor degree.

For MALSTM-FCN, for each graph, time series are created by using following graph features : average clustering coefficient, graph clique number, number of connected components, local efficiency, number of isolates and also normalized time distance to source tweet. We also discarded all edges that point against the time.

4 RESULTS AND DISCUSSIONS

Stratified K-Fold cross validation model (with k=10) is used to measure the performance. For each fold, dataset is split into training (90%), validation (3% of training) and test (10%) sets. Figure 1 shows the results for K-Fold CV by using categorical accuracy, ROC AUC and Matthews correlation coefficient (MCC) and also the official results for test dataset (T-MCC).

Table 1: Stratified K-Fold CV and submission results

Model	Accuracy	ROC AUC	MCC	T-MCC
GCN	71.1± 0.2%	82.6 ± 1.1%	0.56 ± 0.003	0.020
DGCNN	71.2 ± 0.2%	81.5 ± 1.0%	0.56 ± 0.004	0.023
MALSTM-FCN	71.5 ± 1.7%	82.1 ± 1.7%	0.54 ± 0.004	0.035

The results are not promising at all. What went wrong? As the results are really bad, we can not conclude that it was just a problem of tuning. Probably, there is a bug between the code where we train and generate results. It can be an explication for the huge difference between MCC values. We investigate on this.

As a future work, it can be interesting to propose two steps classifier for this task : First to detect fake and not fake by using [8] which should give around 92% ROC AUC and then try to make the distinction between corona and other conspiracy. For the time series part, we would like to focus on this approach by using different features in the future.

REFERENCES

[1] Hunt Allcott and Matthew Gentzkow. 2017. Social Media and Fake News in the 2016 Election. *Journal of Economic Perspectives* 31 (May 2017), 211–236. https://doi.org/10.1257/jep.31.2.211

[2] Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda. 2020. A gentle introduction to deep learning for graphs. *Neural Networks* 129 (Sept. 2020), 203–221. https://doi.org/10.1016/j.neunet.2020.06.006

[3] CSIRO's Data61. 2018. StellarGraph Machine Learning Library. https://github.com/stellargraph/stellargraph. (2018).

[4] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, Gaël Varoquaux, Travis Vaught, and Jarrod Millman (Eds.). Pasadena, CA USA, 11 – 15.

[5] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[6] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. 2019. Multivariate LSTM-FCNs for time series classification. *Neural Networks* 116 (Aug 2019), 237–245. https://doi.org/10.1016/j.neunet.2019.04.014

[7] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]* (Feb. 2017). http://arxiv.org/abs/1609.02907 arXiv: 1609.02907.

[8] Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M. Bronstein. 2019. Fake News Detection on Social Media using Geometric Deep Learning. *arXiv:1902.06673 [cs, stat]* (Feb. 2019). http://arxiv.org/abs/1902.06673 arXiv: 1902.06673.

[9] Konstantin Pogorelov, Daniel Thilo Schroeder, Luk Burchard, Johannes Moe, Stefan Brenner, Petra Filkukova, and Johannes Langguth. 2020. FakeNews: Corona Virus and 5G Conspiracy Task at MediaEval 2020. In *MediaEval 2020 Workshop*.

[10] Soroush Vosoughi, Deb Roy, and Sinan Aral. 2018. The spread of true and false news online. *Science* 359, 6380 (March 2018), 1146–1151. https://doi.org/10.1126/science.aap9559

[11] Zhiguang Wang, Weizhong Yan, and Tim Oates. 2016. Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline. (2016). arXiv:cs.LG/1611.06455

[12] Sung Whan Yoon, Jun Seo, and Jaekyun Moon. 2019. TapNet: Neural Network Augmented with Task-Adaptive Projection for Few-Shot Learning. (2019). arXiv:cs.LG/1905.06549

[13] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An End-to-End Deep Learning Architecture for Graph Classification. *AAAI* (2018).