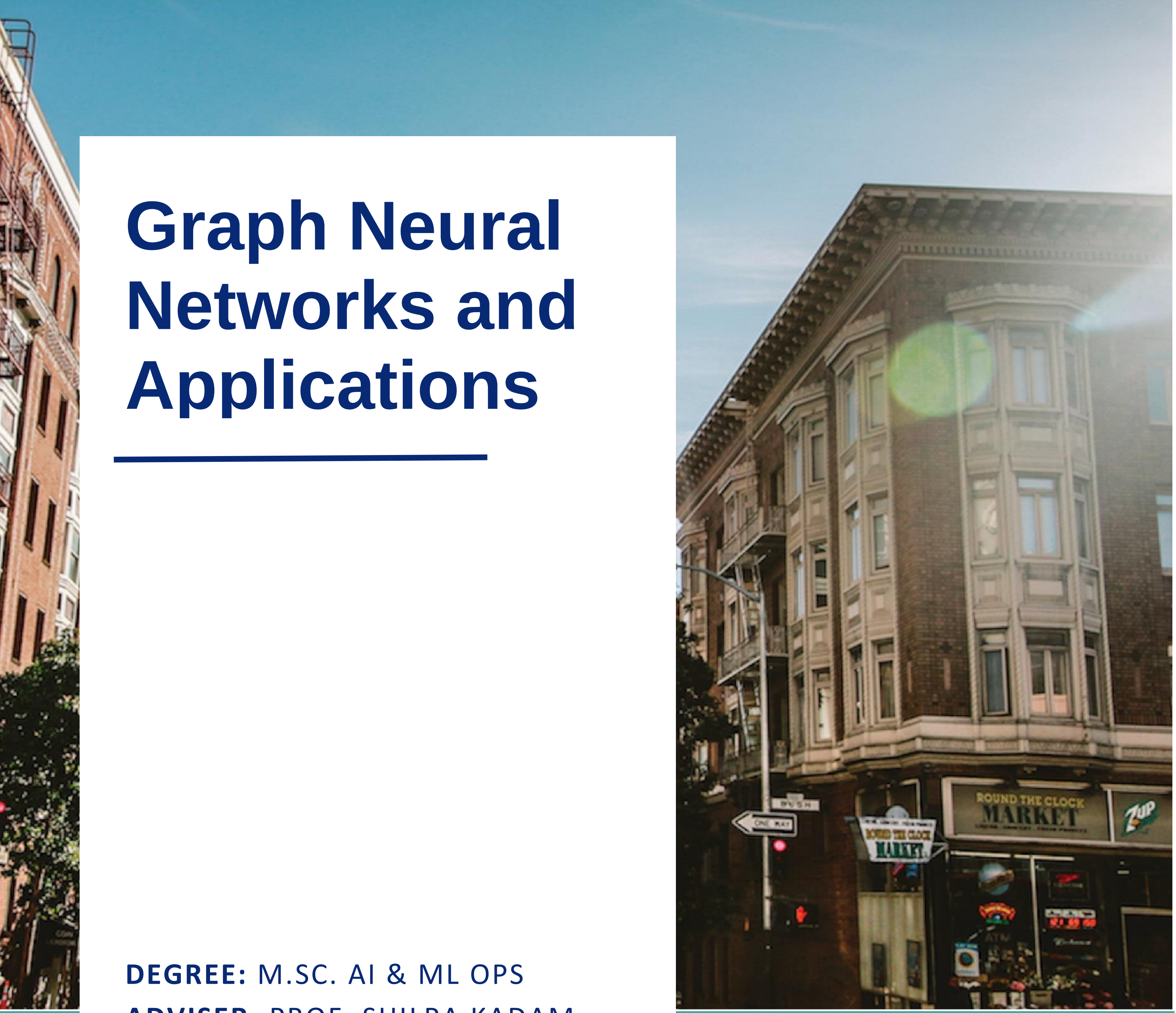


Graph Neural Networks and Applications

DEGREE: M.SC. AI & ML OPS

ADVISER: PROF. SHILPA KADAM

Date: **24th Feb 2023**

MAY 27

JWALIT PATEL

Contents

ABSTRACT:

Graph neural networks have gained much attention in the recent past due to their applications in a wide range of domains, such as molecular structures, social networks, spatial maps, Visual Q&A, object detection, machine translation, and many more.
This project’s scope is to study the fundamentals of graphs, the architecture of graph neural network, their applications, and recent literature work and later apply it to a text classification use case.

Introduction :

What are Graph Neural Networks?

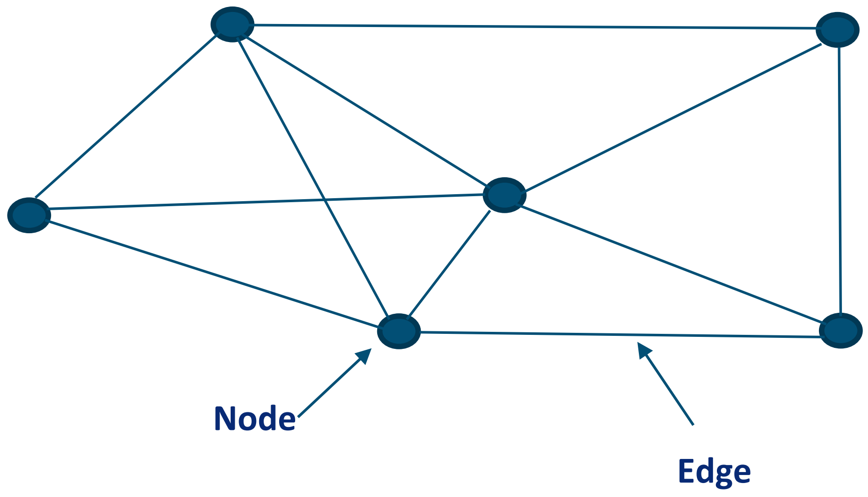


Figure1.1

Real world objects are often defined in terms of their connections to other things. A set of objects consisting connections between them, are naturally expressed as a graph. Researchers have developed neural networks that operate on graph data called graph neural networks.
Many learning tasks require the processing of graphical data containing rich relational information between elements. Modeling physical systems, learning molecular fingerprints, predicting protein interfaces, and classifying diseases all require a model that learns from graphical inputs.

Graph neural networks help address challenges that traditional neural networks have not been able to adequately address. Chart-based data cannot be processed correctly because the connections between the data are not sufficiently weighted. However, for GNNs, the so-called edges are as important as the nodes themselves.

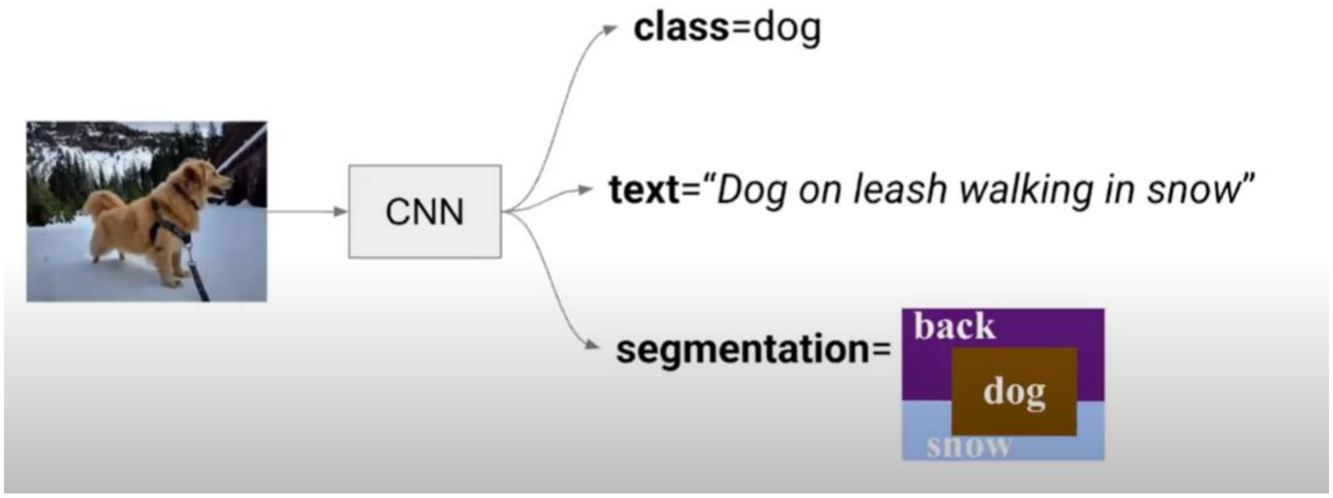


Figure1.2(Source: https://youtu.be/JqWROPYeqjA)

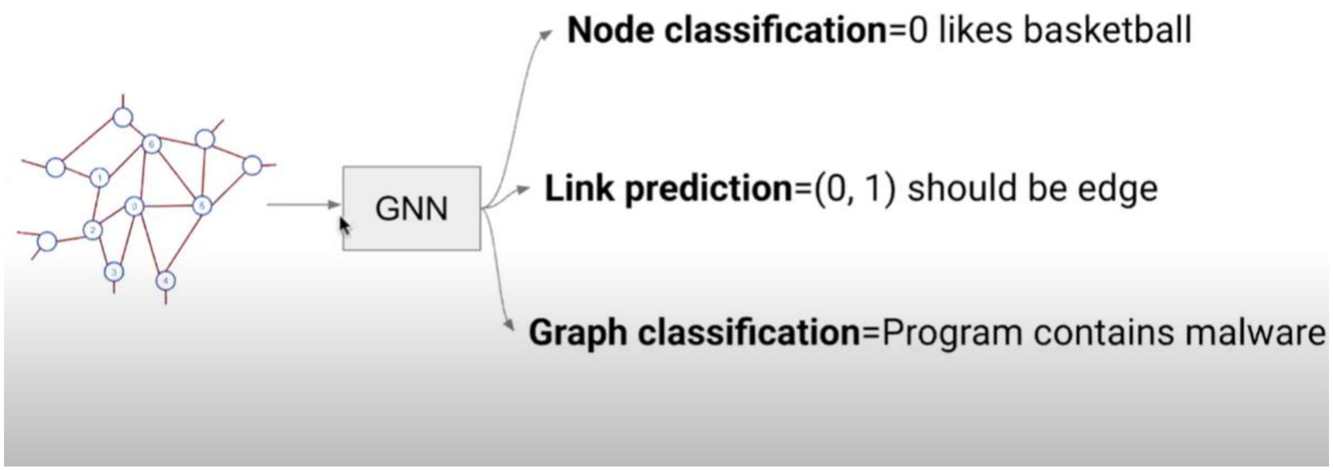

Node classification=0 likes basketball
Link prediction=(0, 1) should be edge
Graph classification=Program contains malware
GNN

Figure1.3(Source: https://youtu.be/JqWROPYeqjA)

GNNs(Graph Neural Networks) typically operate by aggregating information from a node's neighbors to update the node's representation, iteratively propagating information through the graph.
A graph neural network (GNN) is a neural network that can operate on data structured as graphs. A graph consists of nodes (vertices) and edges (links), where nodes represent entities and edges represent relationships between those entities. GNNs are designed to learn functions that map each node of a graph to a fixed-length vector representation.

This representation can then be used in various downstream tasks such as node classification, link prediction, and graph classification.
GNNs have a wide range of applications in various fields.
Some notable applications of GNNs include:

Social Networks: GNNs can be used to model social networks and predict user behavior. For example, GNNs can be used to predict the likelihood of a user following another user or the likelihood of a user clicking on an advertisement.

Natural Language Processing: GNNs can be used to model syntactic and semantic relationships between words in sentences or documents. It can be used for tasks such as text classification, question answering, and machine translation.

Drug discovery: GNNs can be used to predict the properties of molecules and drugs.
For example, GNNs can be used to predict the effectiveness of drugs to treat specific diseases or the toxicity of molecules.

Recommender systems: GNNs can be used to model user-item interactions in recommender systems. For example, GNNs can be used to recommend products to users based on their browsing or purchase history.

Robotics: GNNs can be used to model relationships between objects in a scene and plan robot actions. For example, GNNs can be used to plan the trajectory of a robotic arm based on the positions of objects in the environment.

Overall, GNNs provide a powerful tool for modeling structured data into graphs and have the potential to revolutionize many fields.

Literature Review:

Graph Neural Networks (GNNs) have gained significant attention in the field of Natural Language Processing (NLP) and have been applied to a wide range of tasks such as text classification, sentiment analysis, and named entity recognition. In this literature review, we will focus on the application of GNNs to text classification.

In recent years, several studies have explored the use of GNNs for text classification tasks. One popular approach is to represent the text as a graph, where each node represents a word or token, and the edges represent the relationships between them. This representation allows GNNs to capture the contextual relationships between the words and the structure of the sentence or document.

One of the earliest studies that used GNNs for text classification was the GCN-based text classification model proposed by Kipf and Welling in 2017. The model used a Graph Convolutional Network (GCN) to learn a hierarchical representation of the input text. The authors showed that their model outperformed several baseline models on various text classification tasks.

In survey(William L. Hamilton,2017) makes the use of GraphSAGE, a type of GNN that uses a hierarchical sampling strategy to aggregate information from a node's local neighborhood. The authors demonstrated the effectiveness of GraphSAGE on inductive node classification tasks, achieving state-of-the-art results on benchmark datasets.

The survey(Zonghan Wu,2019) uses Graph Attention Networks (GATs), a type of GNN that uses attention mechanisms to weigh the importance of each node's neighbors. The authors demonstrated the effectiveness of GATs on semi-supervised learning tasks, achieving state-of-the-art results on benchmark datasets.

Following this work, several studies proposed different variations of GNNs for text classification. For example, Zhang et al. proposed a Gated Graph Convolutional Network (GGCN) for text classification, which uses a gate mechanism to selectively update the node features during message passing. The authors showed that their model outperformed several state-of-the-art models on several benchmark datasets.

Another study by Yao et al. proposed a Hierarchical Graph Convolutional Network (HGCN) for text classification, which uses a hierarchical graph structure to capture both the local and global information of the input text. The authors showed that their model outperformed several baseline models on several text classification tasks.
In survey(William L. Hamilton,2017)

Objectives:

1) To study the fundamentals of graphs and study the architecture of graph neural network:

- The basic concept of GNN is message passing. During message transmission, each node sends messages to its neighbors, which are then combined to produce a new representation of that node. Messages can be aggregated through various mechanisms, including message summarization, averaging, or concatenation.
- GNNs consist of multiple layers, with each layer updating the representation of the node based on the output of the previous layer. Representations of nodes can also be updated with information from the graph structure, such as the degree of a node or the number of triangles in which it participates.
- There are several GNN architectures, including Graph Convolutional Networks (GCN), Graph Attention Networks (GAT), and GraphSAGE.

1. GCNs are based on the convolution operation, which is a standard operation in image processing.
2. GAT uses an attention mechanism to weigh the importance of different neighboring nodes when updating the representation of a node.
3. GraphSAGE learns a function that combines a node's own characteristics with those of its neighbors to produce a new representation of the node.

- GNNs have several advantages over traditional neural networks.
- First, they can model non-Euclidean data such as graphs, which are difficult to represent using traditional neural networks. Second, GNNs can handle graphs of different sizes because the message passing mechanism allows nodes to share information with their neighbors.
- Finally, GNNs can handle missing data because nodes can still receive information from their neighbors even if they have no functionality themselves.

Overall, GNNs provide a powerful tool for modeling structured data into graphs and have the potential to revolutionize many fields.

2) To study recent literature work later apply it to a text classification use case.(This also includes approach and method)

- Utilizing the knowledge obtained by the literature review and creating a Fake News Detection using GNN.
- We'll be applying User preference-based fake news recognition which is a powerful method that uses both contextual recommendations and user feedback to identify potential fake news. Taking into account a user's likes and dislikes, this approach can improve the accuracy and efficiency of fake news detection algorithms.
- We'll be applying Fake news Detection using GNN on UPFD dataset using Pytorch Geometric library.
- PyG (PyTorch Geometric) is a library built on top of PyTorch that makes it easy to write and train graphical neural networks (GNNs) for a variety of data generation related applications.There are many methods for deep learning of graphs and other random patterns, also known as geometric deep learning, from a large number of published data. It also includes an easy-to-use minibatch loader, multi-GPU support, torch.compile support, DataPipe support, more general directory (create your own directory) to run multiple small and large images.
- Creating the GCN layers and data input using Pytorch Geometric.
- Training and evaluating the model using TensorFlow Keras and scikit-learn libraries.
- Visualizing graph data using Networkx library.
- NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

DATASET:

It includes two sets of tree-structured fake & real news propagation graphs extracted from Twitter. For a single graph, 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.
The node feature type ("profile", "spacy", "bert", "content").

- If set to "profile", the 10-dimensional node feature is composed of ten Twitter user profile attributes.
- If set to "spacy", the 300-dimensional node feature is composed of Twitter user historical tweets encoded by the spaCy word2vec encoder.
- If set to "bert", the 768-dimensional node feature is composed of Twitter user historical tweets encoded by the bert-as-service.
- If set to "content", the 310-dimensional node feature is composed of a 300-dimensional “spacy” vector plus a 10-dimensional “profile” vector.

In our case we’ll be using word2vec (word2vec is an embedding technique that encodes words into vectors). The nice part about this technique is that the vector space is arranged in a way so that common contexts are located close to one another.

Solution:

Why Graphs

let's begin with developing an understanding for the fake news data set when you think about it fake news detection is not really a graph problem it sounds more like a natural language processing task so why should we even consider using graph based deep learning models for this we can actually combine a language model with a graph model to improve fake news detection based on the retweeting pattern and the preferences of each node we can increase the quality of the predictions what we try to do here is to detect fake news based on structural features as well as language features what we however do not do is fact checking which means comparing the knowledge contained in a piece of news with the real world or to put it differently we don't assess the truthfulness of the content of an article we just look at some patterns and try to predict if this corresponds to a typical fake news article

Notebook / Model

- The first step is to install Pytorch geometric using these commands and then we can get a dataset which in pytorch geometric is called "UPFD" which stands for User Preference Aware Fake News Detection and here you have two options for subset datasets and we will select "gossipcop" here and specify content as a function and that means we use word2vec spatial embedding plus 10 profile functions, so we have a final shape of 310 except that the dataset has already been split in the validation and testing

- Let's examine the edges in this dataset there is one root node, which is connected to many other nodes like in this example the root node is zero and it's connected to one two third four basically different users and these are basically the first retweets and from there the other retweets are based on those users that the original node connected then we convert these data points to a network x object and then we simply use the network x draw function and with that you can have a good look at what the propagation graph looks like so you can also visualize

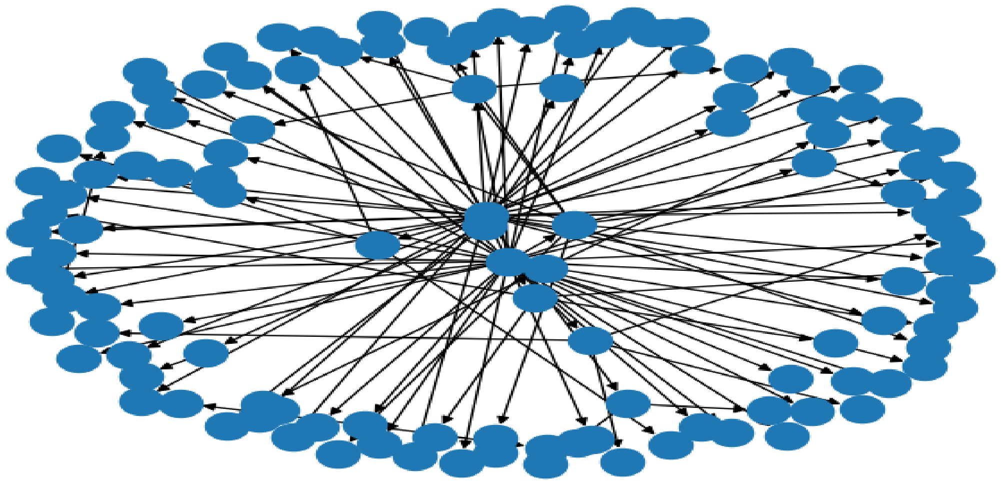


Fig:1.1

- In Fig:1.1 It is a graph consisting 310-dimensional content feature is composed of a 300-dimensional user comment word2vec (spaCy) embedding plus a 10-dimensional profile feature.

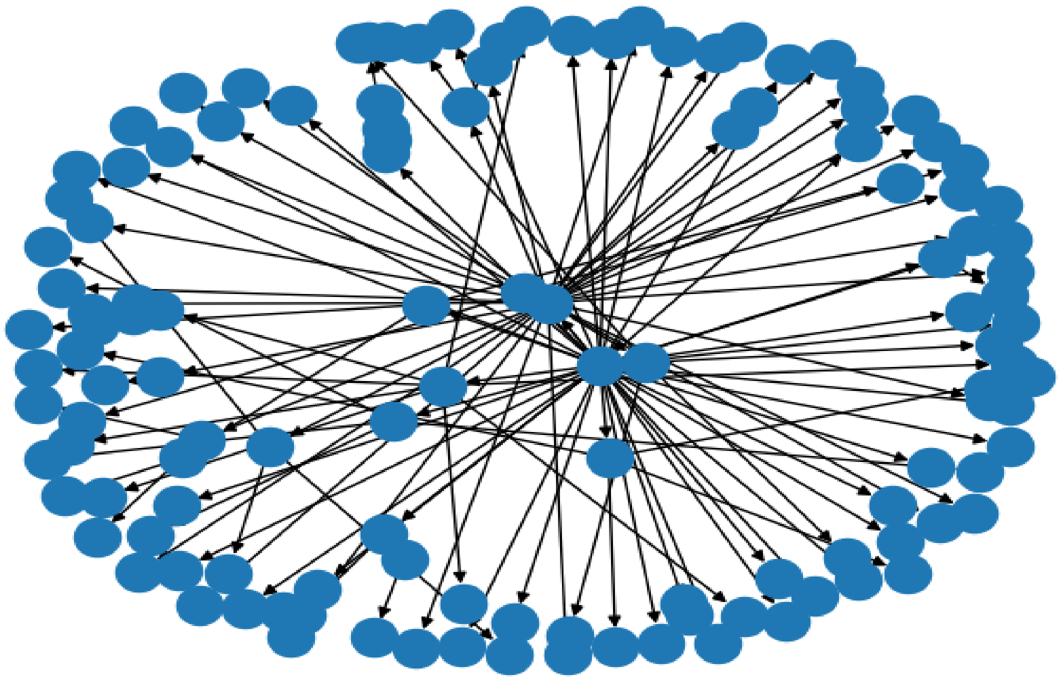
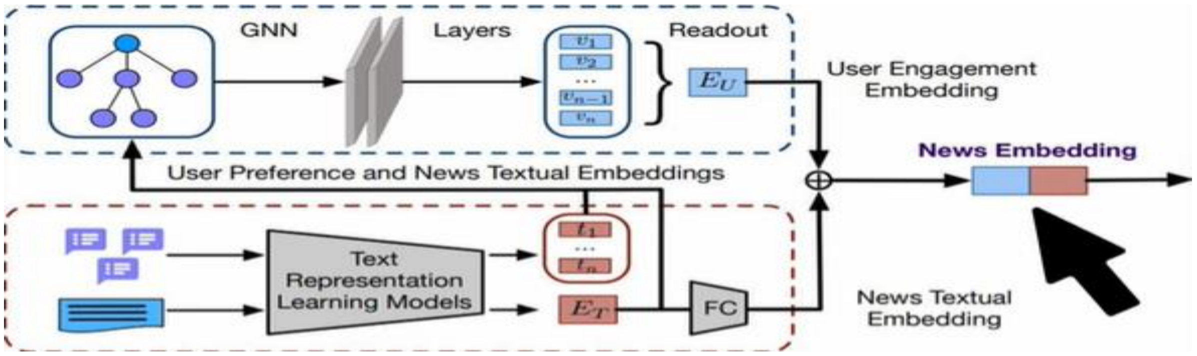


Fig:1.2

- Fig 1.2 is a graph consisting 768-dimensional BERT features are encoded using pretrained BERT.

- Then add these train datasets and test datasets to data loaders from pytorch geometric
- For the model part here we have three (GAT) attention graph layers and several linear transformations that use finite embeddings and then output a single value in the forward pass, we use our graph and features as well as neighborhood information.
- Passing it through these GNN layers which will give us some final embedding in the last layer and after this step we have node embeddings for each of the nodes, but of course we are doing graph classification so we have to somehow collect information about these nodes and here it's a very simple operation pooling which is global maximum pooling
- The Special part model is instead of using only these embeddings they also used raw language embeddings and concatenated that information so that means we use these pooled graph embeddings and combine them with by inserting raw news articles and we have to perform several operations to achieve this and because we just want the message nodes, and to get the node messages.
- There's a bit of a scramble and then we basically get the indexes for the root nodes for the message nodes, and then we just use the raw functions and index them with those indexes root nodes and then do all this with a separate linear transformation .
- Finally these global embeddings are combined with the message embedding and this is sent through another linear transformation and therefore the size is twice the hidden channels here and then we get the output value or the output embedding, which is the joint embedding, and then we pass it through the sigmoid layer to get a value between zero and one yes, it's almost full architecture


GNN
Layers
Readout
v_1
v_2
v_{n-1}
v_n
E_U
User Engagement Embedding
News Embedding
User Preference and News Textual Embeddings
Text Representation Learning Models
t_1
t_n
E_T
FC
News Textual Embedding

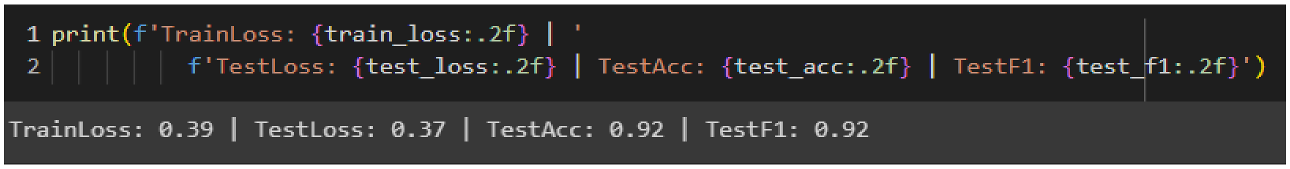
Training and Testing time:

To train a model a trained function a test function and a matrixfunction is used because this is a classification problem we be calculating the accuracy and f1 score and do that inside of the test function and for the train function we just calculate the loss and calculate the loss for which we'll be using (BCE) binary cross entropy loss since is a binary classification problem

For optimizer we'll be using Adam optimizer

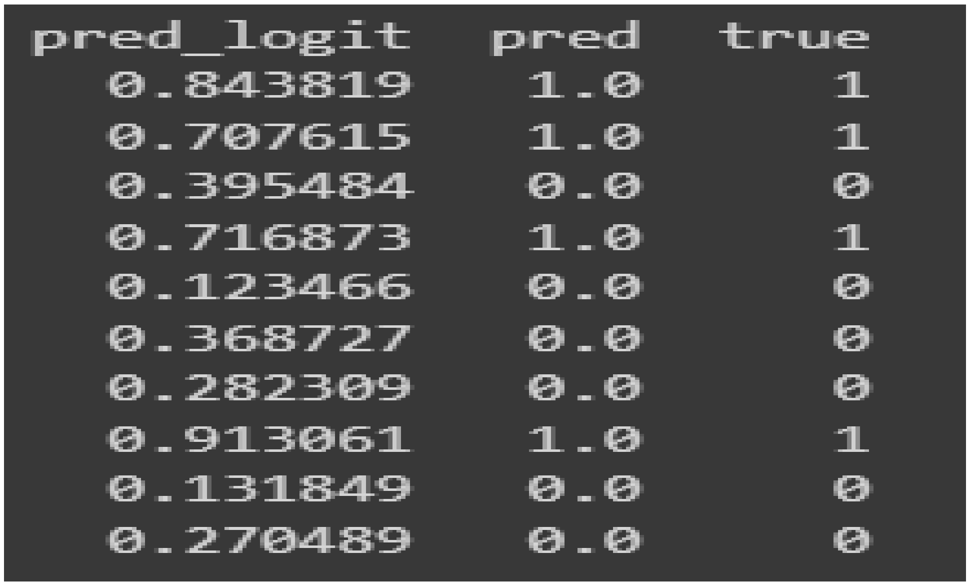
After training we achieve an f1 score and an accuracy of 92 for “content” data whereas for “BERT” data we get an f1 score and accuracy of 69 and 57 respectively and on printing some sample predictions with logic value as well as the actual prediction we get to see that most of the predictions are correct

Output for Content Data



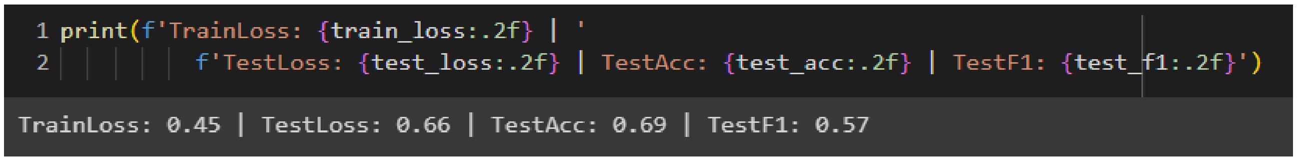
```
print(f'TrainLoss: {train_loss:.2f} | '
      f'TestLoss: {test_loss:.2f} | TestAcc: {test_acc:.2f} | TestF1: {test_f1:.2f}')
```

```
TrainLoss: 0.39 | TestLoss: 0.37 | TestAcc: 0.92 | TestF1: 0.92
```



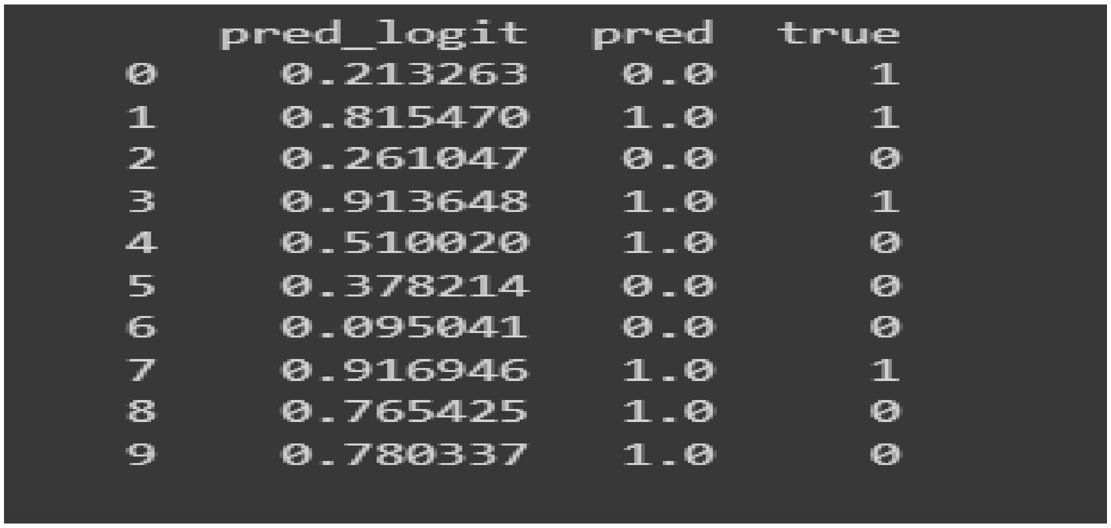
```
pred_logit  pred  true
  0.843819   1.0     1
  0.707615   1.0     1
  0.395484   0.0     0
  0.716873   1.0     1
  0.123466   0.0     0
  0.368727   0.0     0
  0.282309   0.0     0
  0.913061   1.0     1
  0.131849   0.0     0
  0.270489   0.0     0
```

Output for Bert Data



```
print(f'TrainLoss: {train_loss:.2f} | '
      f'TestLoss: {test_loss:.2f} | TestAcc: {test_acc:.2f} | TestF1: {test_f1:.2f}')
```

```
TrainLoss: 0.45 | TestLoss: 0.66 | TestAcc: 0.69 | TestF1: 0.57
```



```
    pred_logit  pred  true
0     0.213263   0.0     1
1     0.815470   1.0     1
2     0.261047   0.0     0
3     0.913648   1.0     1
4     0.510020   1.0     0
5     0.378214   0.0     0
6     0.095041   0.0     0
7     0.916946   1.0     1
8     0.765425   1.0     0
9     0.780337   1.0     0
```

Link:

Link for “BERT” data:

https://colab.research.google.com/drive/1gRvcGjZRb4OynCniVipkdSFIwBZKFnvo?usp=sharing

Link for “content” data :

https://colab.research.google.com/drive/1pStNurg0wlPcGGVdlgEtM_pY_7dDC-k_?usp=sharing

CONCLUSION :

In this paper, we get to know that GNN can be successfully used for NLP task. By comparing both the dataset we learn that while implementing GNN, the higher the number of data in the graph the better the performance it provides.

The Experimental results demonstrates the advantage of using data with news and profile embeddings as it provides more information to the model. In conclusion, GNNs offer promising capabilities for fake news detection by capturing relationships, incorporating contextual information, and leveraging transfer learning. By considering the network structure and interactions, GNNs can provide a valuable tool in combating the spread of misinformation. However, it is important to carefully design and adapt GNN models to address the specific challenges and limitations of fake news detection tasks.

Reference:

1. Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang,
   Changcheng Li, Maosong Sun, Graph neural networks: A review of methods and applications, AI
   Open, Volume 1, 2020, Pages 57-81,ISSN 2666-6510,https://doi.org/10.1016/j.aiopen.2021.01.001.
2. https://paperswithcode.com/paper/graph-neural-networks-a-review-of-methods-and
3. https://paperswithcode.com/paper/how-powerful-are-graph-neural-networks
4. https://github.com/zshicode/GNN-for-text-classification
5. H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, Q. Yang, Large-scale hierarchical text classification with recursively regularized deep graph-cnn Proceedings of WWW (2018), pp. 1063- 1072
6. L. Yao, C. Mao, Y. Luo, Graph convolutional networks for text classification, Proceedings of AAAI, 33 (2019), pp. 7370-7377
7. Y. Zhang, Q. Liu, L. Song, Sentence-state lstm for text representation, Proceedings of ACL, 1 (2018), pp. 317-327
8. K.S. Tai, R. Socher, C.D. Manning, Improved semantic representations from tree-structured long short-term memory networks. Proceeding of IJCNLP (2015), pp. 1556-1566
9. Attention-based Graph Neural Network for Semi-Supervised Learning by Zonghan Wu et al. (2019)
10. GraphSAGE: Inductive Representation Learning on Large Graphs by William L. Hamilton et al. (2017)
11. https://github.com/stellargraph/stellargraph
12. https://paperswithcode.com/paper/user-preference-aware-fake-news-detection
13. https://www.youtube.com/watch?v=QAIVFr24FrA
14. https://pytorch-geometric.readthedocs.io/en/latest/tutorial/compile.html