UNIVERSITY OF FRIBOURG

BACHELOR THESIS

Thesis Title

Author:
Mattias Dürrmeier

Supervisor:
Prof. Dr. Philippe Cudré-Mauroux

Co-Supervisor:
Co-supervisor Name

May 19, 2023

eXascale Infolab
Department of Informatics

Boulevard de Pérolles 90 • 1700 Fribourg • Switzerland
phone +41 (26) 300 84 65 • diuf-secr@unifr.ch • www3.unifr.ch/inf

Abstract

Mattias Dürrmeier

Thesis Title

Write the thesis abstract here. Should be between half-a-page and one page of text, no newlines.

Keywords: keywords, list, here

Contents

List of Figures

Chapter 1

Dataset

1.1 Misinformation Dataset

FakeNewsNet is a dataset collected from the Twitter social media platform, made of political *tweets*. The original dataset along with its paper was first published in September of 2018. In this thesis, we specifically work with a pre-processed subset of this dataset, made available alongside the Factual News Graph (FANG) source code. We will refer to this datset as the misinformation dataset.

This misinformation dataset is a heterogeneous network: the vertices are not all of a same and unique type. An entity can be of type user, news article or news source. Edges can be between entities of the same or different type. This dataset contains over 3 millions edges, named interactions in this context.

TABLE 1.1: Statistics about the misinformation network.

$\|V\|$	55957
$\|E\|$	3242492
Labels	truth value

The entity news articles is a central part of this dataset: it is the channel through which users and news sources interact with one another. The users post and comment on a news piece via *tweets*.

Each news article can be either fake or real. For each news, the corresponding label was collected from the websites PolitiFact and Snopes, two popular fact checking websites mainly focusing on U.S. politics. These websites were created in an attempt to fact check information and political statement made from news, sources and politicians.

TABLE 1.2: Statistics about the network entities.

Fake news	448
Real news	606
Source	442
Users	54461

Interactions can be separated in two categories: those with a single label, between sources, news and users and those with multiple labels, between a user and a news article. For the first category, we have the following label alongside the interaction it represents:

- citations, between source and source. A news source can cite other news sources in an article.

- relationship, between user and user. On Twitter, a user can interact with other users, through replies and mentions on a *tweet*.
- publications, between source and news article. News source process and publish information through news articles.

The second group of interactions, *stance*, exclusively concern user and news article. Here, multi-labels is necessary because users can have different reaction and attitude towards a piece of information. For this group, we have the following labels and their meaning:

- negative support, a negative reaction supporting the news article statement. A user agrees with a piece of information and reacts in a negative manner towards it.
- neutral support, a neutral reaction supporting the news article statement. A user agrees with a piece of information but reacts in a neutral manner.
- deny, a reaction denying the news article statement. A user denies a statement from a piece of information.
- a report, a verbatim reporting of the new article. A user reposts the information using similar words from the original news article.

The dataset also has timestamps for each *tweet*. This metric is useful to measure the amount of reactions a news article receives through time.

1.2 FANG: Available Data

In this section, we take a look at the available raw data in FANG's dataset. The repository provides some of the raw data and a processed version of the dataset. The code to process the data is not made available by the authors. In an attempt to recreate the preprocessed misinformation dataset, we wanted to know how much of the original data we could retrieve back today in 2023.

FANG's author link the FakeNewsNet code to crawl the unprocessed data. In its raw, unprocessed form, FANG's data is separated in two comma separated values (CSV) files: one for fake news article and one for real ones. Each CSV provides a list news id, URL to the news source, title of the news article, and a list of unique *tweets* which took a stance towards the news article. Table 1.3 illustrates a single line of this raw data.

TABLE 1.3: Example of raw data for a real news article.

news id	pheme_144
news URL	`https://www.reuters.com/article/us-australia-security/...`
title	"Hostages held in Sydney cafe, Islamic flag seen in window: local TV"
User list	{ 544289893401522177, 544281840358801408, 544289496758362113, ...}

We technically could scrap some of the data. We can get back the users' original *tweet* that interacted with a news piece using the unique tweet ID. However recent changes in Twitter's API policy means we cannot crawl Twitter for the *tweets*. This means we cannot retrieve the user stances towards a news article.

We must then evaluate the user stance towards the news article. To tune the sentiment, FANG's authors used a large scaled pre-trained model named ROBERTa. They trained the model on the Yelp Review Polarity dataset to further classify neutral and negative sentiment. Pre-trained model such as ROBERTa are now available for download online on model sharing platform. The Yelp Review Polarity dataset is also widely available. It would be therefore possible to download the model and the dataset and train the model to evaluate the user stance.

To get the additional data crawled for the media sources, we could use the URL for each news article. By visiting each of the news source website, we can crawl the *Homepage* and *About Us* sections. The code to crawl the news source is not provided with FANG; we would have to build our own crawler.

With some effort, some part of the dataset could be reconstructed. However scraping and reconstructing the dataset is outside of the scope of this bachelor thesis. We will therefore rely on the preprocessed version of the dataset made available with FANG.

1.3 From Heterogeneous to Homogeneous

Deepwalk and node2vec, the algorithms discussed later in this chapter, can only be used with homogeneous graph. Because our network is heterogeneous, we must transform it by removing the entities types.

We obtain a transformed homgeneous network by assigning a unique index to each entity, no matter the type, in the entity list of the network. We then map for an entity the corresponding index to each vertices in the graph. This results in the same network G but homogeneous.

We then build the graph from the resulting homgeneous edgelist and verify that each entity is present in our resulting graph. This graph greatly simplifies computing the adjacency list and edgelist for each entity, lists used as input data for the DeepWalk and node2vec algorithm.

Chapter 2

Experiments

2.1 DeepWalk

Deepwalk is an algorithm to learn latent representation in a network. It is the first of its kind to take successful techniques usually reserved for natural language processing and apply them to network analysis.

To learn the features and relations between vertices, the algorithm walks randomly through a network. Deepwalk is a form of unsupervised learning: the model learns independently from the labels, if available. This makes it a powerful algorithm for finding latents representation in very large networks even with sparse data.

Deepwalk is based primarily on the SkipGram model, often referred to as *word2vec*. *Word2vec* is a technique created for natural language processing published in 2013. It is used to perform word embeddings: in a higher dimensional space, words of similar meaning and used in similar context are grouped together, away from the words used in a different context. Here, Deepwalk applies this technique to learn the relationship between nodes in a network. The idea behind it is to treat a network as a document. Since we're working with network, the SkipGram model is trained with each random walk performed. In natural language processing, *word2vec* represents words with similar meaning close to another in the resulting embedding space; in Deepwalk, this will be the case for nodes with similar features.

Deepwalk has four main hyperparameters which can be tuned to obtain a better representation of the network. We can choose the size of the context of a SkipGram model: in NLP, this is the number of words included before and a after a word to then chose the next one. With a network, this will be the number of neighboring vertices to take into account for the next random walk.

In addition, the output embedding can be tuned via the embedding size d. This is the number of latent dimensions we want to have for a graph. A higher or lower dimensionality can result in a better representation of the network in the embedding space. However more dimensions does not necessarily mean better results. It is important to find the right hyperparameter which leads to the best results.

We can set the number of random walk started at each vertex with the parameter γ, and the length of each random walk with the parameter t. Through these parameters, we can tune DeepWalk's random walk behaviour. Depending on the network, we might prefer DeepWalk to do numerous short walk or few long ones.

2.1.1 DeepWalk With Disinformation Dataset and Parameters Sensitivity

We used DeepWalk algorithm on our disinformation dataset. For this experiment we perform the graph embedding on the adjacency list discussed in subsection 1.3. In a text, more context helps the algorithm to better understand the meaning of a word and how to place them in the embedding space. Similarly, more vertices helps gain information on the network underlying function. For this reason, we use the entire network data to perform the embedding.

The resulting embedding representation is then cleaned from users and sources entities so that we score only the news articles. To transform the network to homogeneous, we assigned a unique index for each entity and mapped it to the graph's edge list. Conveniently, we can reuse this mapping to now remove all indexes which are not news entities.

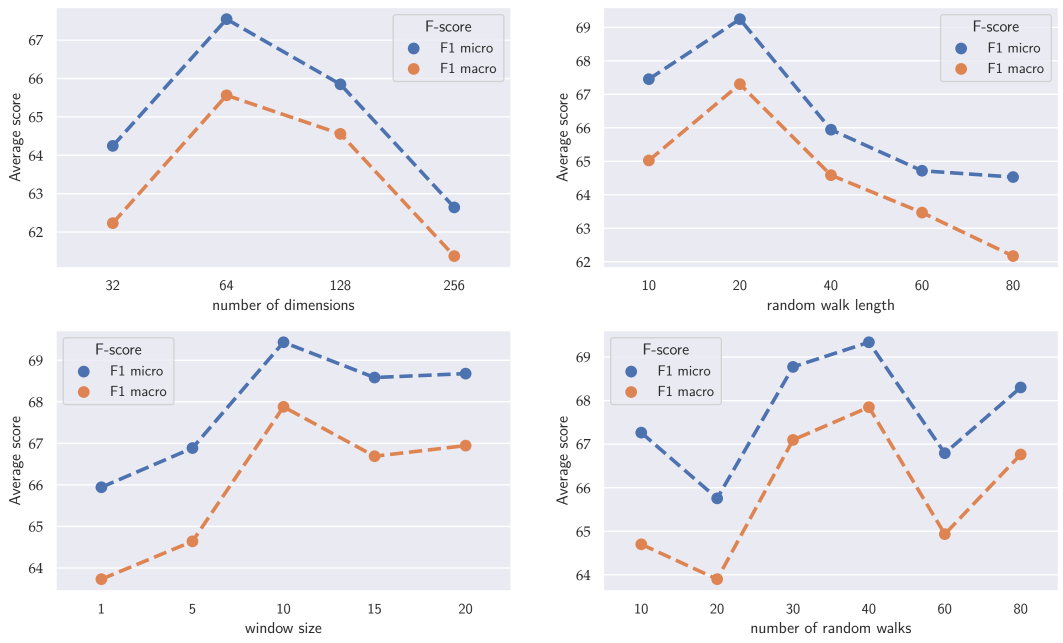

FIGURE 2.1: DeepWalk results.

A new classification model is trained on the cleaned embedding to classify fake and real news. We use the news labels provided with the dataset as targets. Ideally news should be grouped together by veracity in the resulting embedding space since they share similar characteristics. Here, we want to test if DeepWalk was able to group them well.

To obtain the best performance, each hyperparameter of the algorithm is tuned one after the other. We first tune the number of dimensions d, with all the other parameters set to their respective default. We select the number of dimensions which resulted in the best F-score. This value is then used for all remaining parameters tuning. This process is repeated for the w, γ and t parameters in this order. We always use the best parameter value for the parameters previously tuned, and leave the other left to tune to their default values.

Figure 2.2 showcases the F-score obtained for all 4 parameters tuning.

For DeepWalk, the best parameters are $d = 64, w = 10, \gamma = 20, t = 40$. For the misinformation network, DeepWalks prefer few but quite short random walks with an average number of vertices to chose from.

2.2 Node2vec

The node2vec algorithm uses a similar embedding approach as DeepWalk: random walks through the network to learn latent features.

However node2vec's authors found that purely random walk sometimes did not give the best possible result. Indeed if the walks are purely random, we could obtain very different embeddings between two instances of DeepWalk on the same network. For this reason, they decided to add two additional hyperparameters: tuning for breadth first search and depth first search. These two concepts can be tuned through the return parameter denoted p and in-out parameter denoted q. These two parameters are probability and influences the exploration of the network. They are used as $1/p$ and $1/p$ by the algorithm in the graph.

The return parameter controls how often the algorithm revisits a node previously visited in the network. Thanks to this parameter, we can tune the random walks behaviour. A high value for this parameter makes it less likely we revisit a node and encourages exploration. While A low value ensure the algorithm walks closely to the starting node.

The in-out parameter controls the probability to visit not yet visited neighbours. A high value will encourage exploration away from the neighborhood while a low value will lead to a more conservative walk on the graph.

2.2.1 Node2vec with Disinformation Dataset and Parameters Sensitivity

Just like for Deepwalk, we again perform the embedding with node2vec on the integrity of the network to gain from the additional context provided by users and news source. We always clean the resulting embedding, since we want to perform the classification on the news entity only.

Before tuning the four hyperparameters for the output embedding, we must first tune the p and q parameters. We obtained embeddings by iterating over these two parameters with a set of 5 values for each, for a total of 25 runs. All other parameters are set to their default. Table 2.1 shows us the micro F1 score obtained by tuning these parameters. We obtain $p = 2$ and $q = 1$ as the best parameters for the node2vec with the fake news dataset.

TABLE 2.1: Micro F1 score for p and q.

p \ q	0.25	0.5	1.0	2.0	4.0
0.25	0.6564	0.6720	0.6550	0.6583	0.6564
0.5	0.6493	0.6635	0.6687	0.6408	0.6450
1.0	0.6649	0.6848	0.6536	0.6640	0.6588
2.0	0.6829	0.6716	**0.6943**	0.6744	0.6654
4.0	0.6749	0.6739	0.6773	0.6540	0.6815

Just like for DeepWalk, when then tune each four hyperparameters so we can obtain the best possible results for this algorithm on our dataset. We first tune the number of dimensions and leave all other parameters left to tune to their default. We pick the parameter value for the number of dimensions which gave the best F1 score and use this value for all next runs. The parameters which gave the best F1 score for each run is picked for the tuning of the following parameters. We continue doing this for the window size, the random walk length and the number of random walk.

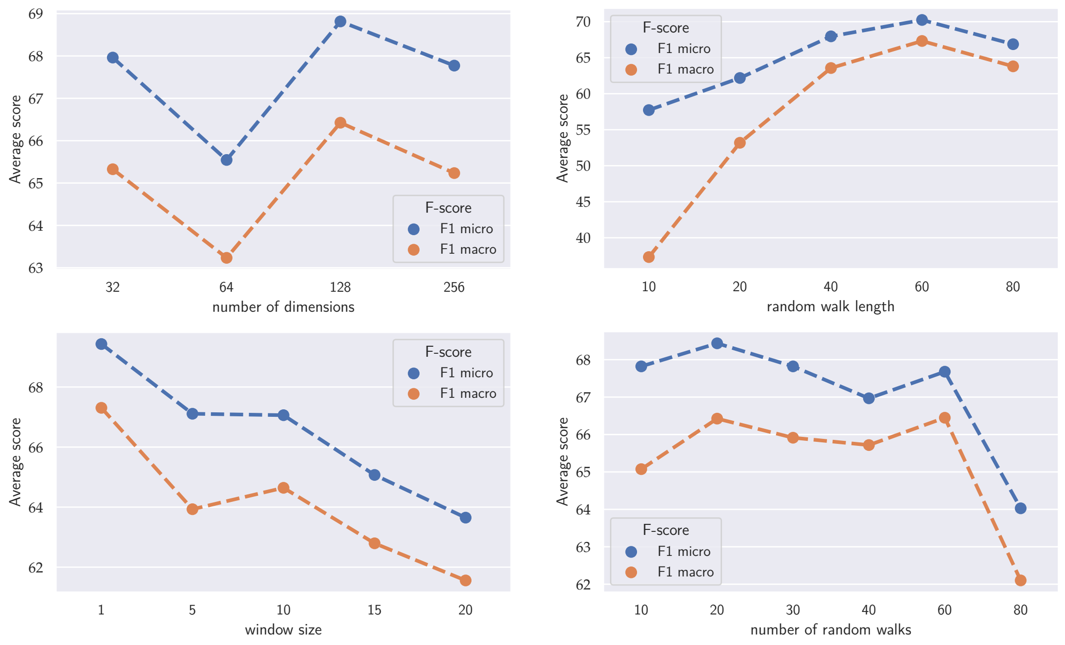

FIGURE 2.2: Node2vec results.

2.3 DeepWalk and Node2vec Results Discussion

The parameters which resulted in the best score for node2vec on our disinformation dataset are $p = 2, q = 1, d = 128, w = 1, \gamma = 60, t = 20$.

for node2vec, the best window size parameter was 1, compared with 10 for DeepWalk. This means that node2vec consider only one possible node for the next step of the random walk. We believe this strong difference in window size between the two algorithm could be due to the additional p and q hyperparameters of node2vec. With a value of $p = 2, q = 1$, node2vec prioritize exploration over revisiting known nodes. Because our graph is highly connected, network exploration means fewer but longer random walk, as we can see from $\gamma = 60$ and $t = 20$ for node2vec. The hyperparameters $\gamma = 20$ and $t = 40$ shows that DeepWalk prioritize network exploration through randomness, with shorter but more numerous walks.

Compared with DeepWalk, we can a see slightly better performance improved by node2vec. DeepWalk strictly performs the network exploration randomly. Node2vec has a small edge here, because its walks can be influenced through p and q parameters. In a large network such as the misinformation dataset we use here, this improves the graph exploration slightly. In our best result, we prioritize exploration of unknown nodes on the network over already visited ones. This leads to a slight but significant improvement in the algorithm's network knowledge and thus a better representation in the resulting embeddings.

2.4 Score Comparsion

With our two algorithms best performance, we can now compare their performance compared to FANG. In FANG's original paper, its performance was measured with Area Under the ROC Curve, AUC for short. Conveniently FANG also produces a F1 score when we perform it on our dataset. We will therefore use both metrics to compare the performance of these algorithms.

We measured the performance of Deepwalk and node2vec on an embedding performed with the best parameters, discussed previsouly in chapter 2. For FANG, we enabled all hyperparameters such that it uses all available context, therefore obtain the best performance possible.

TABLE 2.2: Comparison between models, evaluated with F1 score and AUC.

Model	F1 score (micro)	AUC
DeepWalk	0.6810	0.6493
node2vec	0.6844	0.6654
FANG (10%)	0.5828	0.6479

We can now compare these metrics of both algorithm with the one from Factual News Graph (FANG).

Chapter 3

FANG

3.1 Hyperparameters

FANG has four hyperparameters which can be either activated or deactivated: temporality, stance, proxmity and attention.

Table 3.1lists the AUC scores obtained with FANG trained with 80% of the data. We otain these scores by running FANG on the misinformation dataset explored in chatpter 1. We progressively activated the hyperparameters one by one, until all where fully activated.

TABLE 3.1: FANG AUC scores with some hyperparameters combination.

Attention	Stance	Proximity	Temporality	AUC Score
✓				0.0
✓	✓			0.0
✓	✓	✓		0.0
✓	✓	✓	✓	0.0
✓	✓	✓	✓	**0.0**

3.2 FANG Test Run on CPU

Typically neural network model of this complexity and size are trained on GPU.

For testing, a CPU flag was created: this allows to train FANG on a machine that only has a CPU. All possible flags were activated (attention, stance, proximity and temporality). For this test run, only 10 percent of the data was used. This is because training on CPU is much much slower compared to GPU.

```
python run_graph.py \
    -t fang \
    -m graph_sage \
    -p data/news_graph \
    --percent 10 \
    --epochs=30 \
    --attention \
    --use-stance \
    --use-proximity \
    --temporal \
    --use_cpu
```

We obtained good results for a first run on only 10 percent of the data. The F1 score is 0.58. This result is similar to DeepWalk with the best parameters at 10 percent of the training data. However we expected to see FANG's score value climb much faster than DeepWalk and node2vec once the percentage of training data is higher.

```
    Number of test: 465
    Accuracy score: 0.6129032258064516
    Precision score: 0.6021869620867724
    Recall score: 0.5872625508819539
    F1 score: 0.5827517447657029
    ROC AUC score: 0.647859188903965
    {'accuracy': 0.6129032258064516,
    'precision': 0.6021869620867724,
    'recall': 0.5872625508819539,
    'f1': 0.5827517447657029,
    'auc': 0.647859188903965,
    'loss': 0.702542781829834}
```

3.3 Code Structure

This is a temporary section, just to have notes when we discuss the code.
FANG has the following flags to use:

- the task: fang or news graph.
- the model: graphSAGE or GCN. GraphSAGE can only be used with FANG, and GCN only with news graph.
- percent: Percentage for the size of the training set. 10, 30, 50, 70, 90. Hardcoded in JSON files.
- temporal: whether to use temporality or not. Default to false.
- stance: whether to use stance or not. Default to false.
- promixity: whether to use proximity. Default to false.
- attention: whether to use attention. Default to false.

In terms of directories and file:

- data/: directory for the FANG dataset. Processed and unprocessed.
- dataset/: for the NewsGraph class. This is to run news graph instead of FANG.
- fang/: everything related to graphSAGE: implementation, building the graph, loading the data in the graph. Also FakeNewsClassifier and StanceClassifier classes.
- graph/: GCN and NGCN. The code only uses NGCN.
- training/: for the training of FANG.
- use_embed/: for embedding the users in graphSAGE.
- run_graph.py: this is where FANG begins.

code flow:

- start in run_graph.py. We can either run news_graph or fang.
- news_graph goes into `training/news_graph.py`. The required model is GCN. It build a NGCN as model. It trains the model, validates and tests it.
- fang requires the model to be graphSAGE. The program goes to `fang/graphsage/`. It loads the dataset and a lot of parameters.
- It build the FANG graph, create a StanceClassifier and a FakeNewsClassifier. It starts training, testing and validation.