Content based methods for misinformation detection

Adrien BENAMIRA
CentraleSupélec
adrien.benamira@supelec.fr

Benjamin DEVILLERS
CentraleSupélec
benjamin.devillers@supelec.fr

Etienne LESOT
CentraleSupélec
etienne.lesot@supelec.fr

Ayush K. RAI
CentraleSupélec
ayush.rai2512@student-cs.fr

Manal SAADI
CentraleSupélec
manal.saadi@supelec.fr

Abstract

Because of Fake News proliferation, their detection is an emerging problem that has become extremely prevalent during the last few years. Most of the existing methods on this topic focus on manual feature extraction and supervised classification models leveraging a large number of labeled (fake or real) articles. In contrast, we focus on content-based detection of fake news articles, while assuming that we have a small amount of labels, made available by manual fact-checkers or automated sources. We argue that this is a more realistic setting in the presence of massive amounts of content, most of which cannot be easily fact-checked. This work proposes a text classification approach with small amount of labeled data using a graph neural network method based on attention. The method is twofold. First, we reproduced the results of [12] AGNN algorithm on standard benchmarks. We did two different implementations: one using the PyTorch-Geometric library, the other one using PyTorch and based on the source code of [7]. Then, we applied AGNNs on a fake news graph built as in [5]. We evaluate our method for multiple embeddings and hyperparameters and compare our results with the method proposed in [5]. It appears that our method is more stable and outperforms [5] by 10% when we only used 10% of labeled data.

1. Introduction

Social networks are becoming the main platforms to spread information. This, with the increasing number of users, makes it easy for fake news creator to fool people into believing in falsified content. Moreover, fake news take advantage of the echo chambers phenomenon amplified by social networks: people tend to follow and share only articles they believe in and what their friends share and like. This is why social media such as Twitter are especially vulnerable to the propagation of fake news mostly coming from unverified publishers and crowd-based content creators. There is practically no control over the information that is shared.

Moreover, in the context of expansion of data and proliferation of news, only a little subset of the whole amount of data will be labeled. However, most existing method on this topic focus on manual feature extraction and supervised classification models leveraging a large number of labeled (fake or real) articles. We believe that this approach is not realistic because we will never be able to label enough data to treat in real time all articles posted on social media. In order to tackle the fact that labels are often very limited and sparse, we want to have a semi-supervised learning method for text classification. Graphs seem to be a very good structure to address such a problem because it provides pairwise relations among the data points, both labeled and unlabeled. The goal of such graph-based semi-supervised learning problems is to classify the nodes in a graph using a small subset of labeled nodes and all the node features. It has recently been demonstrated that the existing approaches can be significantly improved upon on a number of standard benchmark datasets by using an innovative neural network architecture on graph-based data known collectively as Attention-based Graph Neural Network [12].

Several authors tackle the problem Fake News detection with different methods:

- **Content based** where the goal is to classify fake news only based on the content of the article [5].
- **Propagation based** where the goal is to classify fake news based on how it is propagated [2].
- **Source based** where we try to classify fake news base on the source: who published?, who shared?, retwitted?... [13].

Our focus will be on content-based detection methods of fake news articles, while assuming that we have very few labels. An application of this work can be a web browser extension which gives a fake news meter.

2. Problem Definition

2.1. Fake News Model

The method to model our problem is twofold. We use CP/PARAFAC tensor decomposition to embed our articles then build a k-nearest neighbour (knn) graph.

2.1.1 CP/PARAFAC Tensor decomposition

A tensor is a multi-dimensional array where each dimension represents a mode. Canonical Polyadic (CP) or PARAFAC decomposition is a tensor decomposition method, widely used that factorizes a tensor into a sum of rank one tensor. In our case, we will be working on a three mode tensor. It can be decomposed as follows:

$$\sum_{k=0}^{n} a_r \circ b_r \circ c_r$$

Where $a_r \in \mathbb{R}^I$ and $b_r \in \mathbb{R}^J$ and $c_r \in \mathbb{R}^K$ and the outer product given by :

$$(a_r, b_r, c_r)(i,j,k) = a_r(i) b_r(j) c_r(k)$$

for all i, j, k.
The factor matrices are defined as :

$$\begin{aligned} A &= [a_1, a_2, \cdots, a_R] \\ B &= [b_1, b_2, \cdots, b_R] \\ C &= [c_1, c_2, \cdots, c_R] \end{aligned} \tag{1}$$

Where $A \in \mathbb{R}^{I\times R}$ and $B \in \mathbb{R}^{J\times R}$ and $C \in \mathbb{R}^{K\times R}$ denote the factor matrices and R is the rank of the decomposition, that is, the number of columns in the factor matrices.

2.1.2 k-nearest-neighbour graph

For each node, we calculate the l_2 distance between the node and its neighbours, we rank them, then we select the k closest ones. The l_2 distance between node p and q is defined as follows:

$$d(p,q) = \sqrt{\sum_{i=0}^{n} (q_i - p_i)^2}$$

2.2. Fake News Detection

We used the AGNN model described in the following session and we aimed at minimizing the cross entropy loss $\mathcal{L}$ introduced in the section 3.3.

3. Related Work

3.1. Tensor Embeddings

In order to model the graph of news articles, we applied the methodology described in [5]. The more detailed process is described in the problem definition.

3.2. Graph Neural Networks (GNNs)

Initially, GNNs were introduced as an extension of Recurrent Neural Networks (RNNs), GNNs apply recurrent layers to each node with additional local averaging layers [4], [11]. It could be interpreted as extensions of convolutional neural networks on a 2D grid to general Graph Convolutional Network (GCN). The goal of GNNs are to find a model so that: $Z = f(X, A) \in \mathbb{R}^{I\times d_y}$ that predicts at each node one of the d_y label classes. Z_{ic} is the estimated probability that the label at node $i \in [n]$ is $c \in [d_y]$ given the features X and the graph A.

The data features $X \in \mathbb{R}^{I\times d_x}$ has at each row d_x features for each node, and $A \in \{0,1\}^{n\times n}$ is the adjacency matrix of G.

The forward pass in a typical GNN alternates between a propagation layer and a single layer perceptron. Let t be the layer index. We use $\tilde{H}(t) \in \mathbb{R}^{n\times d_h}$ to denote the current (hidden) states, with the i-th row $H_i^{(t)}$ as the d_h dimensional hidden state of node i. A propagation layer with respect to a propagation matrix $P \in \mathbb{R}^{n\times n}$ is defined as:

$$\tilde{H}^{(t+1)} = PH^{(t)}$$

Next, a single layer perceptron is applied on each node separately and the weights $W^{(t)}$ are shared across all the nodes:

$$H^{(t+1)} = \sigma\left(\tilde{H}^{(t)} W^{(t)}\right)$$

Where $W^{(t)} \in \mathbb{R}^{d_{h_{t+1}}\times d_{h_{t+1}}}$ is the weight matrix and $\sigma(\cdot)$ is an entry-wise activation function.

3.3. Graph Convolutional Network (GCN)

Graph Convolutional Network (GCN) [7] have achieved good performance in benchmark citation networks. GCN is a special case of GNN which stacks two layers of specific propagation and perceptron:

$$\begin{aligned} H^{(1)} &= \text{ReLU}\left((PX)W^{(0)}\right) \\ Z = f(X,A) &= \text{Softmax}\left(PH^{(1)}W^{(1)}\right) \end{aligned} \tag{2}$$

with a choice of $P = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$, where $\tilde{A} = A + I$, where I is the identity matrix, $\tilde{D} = \text{diag}(\tilde{A}\vec{1})$ and $\vec{1}$ is the all ones vector.

The weights are trained to minimize the cross-entropy loss over all labeled examples L:

$$\mathcal{L} = -\sum_{i \in L} \sum_{c=1}^{d_y} Y_{ic} \log(Z_{ic})$$

The problem with GCN is the fact that they are weight-consuming which is critical for semi-supervised learning where the number of labeled examples is small. Besides, there is a lack of interpretability.

3.4. Graph Linear Network (GLN)

GLN is similar to GCN but without the intermediate non-linear activation units :

$$Z = f(X, A) = \text{Softmax}\left((P^2 X) W^{(0)} W^{(1)}\right)$$

GLN achieves an accuracy comparable to the best GNN, and sometimes better. This suggests that the strength of GNN is in the propagation layer and not in the perceptron layer and the propagation layers is critical in achieving the desired performance which leads to the proposal [12] in replacing the propagation layer of GLN with an attention mechanism and test it on the benchmark datasets.

3.5. Attention-based Graph Neural Network (AGNN)

[12] proposes a novel graph neural network that removes all the intermediate fully-connected layers, and replaces the propagation layers with an attention mechanism which respects the structure of the graph. The attention mechanism allows us to learn a dynamic and adaptive local summary of the neighbourhood to achieve more accurate predictions. This proposed attention-based graph neural network captures this intuition and:

1. greatly reduces the model complexity, with only a single scalar parameter at each intermediate layer ;
2. discovers dynamically and adaptively which nodes are relevant to the target node for classification ;
3. improves upon state-of-the-art methods in terms of accuracy on standard benchmark datasets : CORA, PUBMED, CITESEER.

4. Methodology

News articles do not come as a graph structure naturally. In order to use graph-based algorithms for our fake news problem, we first need to construct a graph out of our dataset. Our approach is based on [5]. Then, we apply an AGNN approach to classify all the articles [12]. It is done in three steps: document embedding, k-nearest neighbours, and AGNN learning.

4.1. Graph Design

Given a collection of news articles $\mathcal{N} = \{n_1, n_2, n_3, \cdots, n_M\}$ of size M , where each news article in $\mathcal{N}$ is a vector that contains the words within the news article. The graph $\mathcal{G} = \{E, V\}$ will be designed such that the set of vertices E will contain the articles and we will consider that a vertex $(a_1, a_2) \in V$ connects two articles if the two articles are "close" in some way. The vocabulary is also filtered with respect to the word frequencies, that is to say we delete the most common words.

4.1.1 Embeddings and Node Features

To define this notion of "distance", we will embed an article into a vector and use a simple Euclidean distance between the two articles.

- **Co-occurence Matrix and CP/PARAFAC tensor decomposition** [5]. We build a three-mode tensor $X \in \mathbb{R}^{I \times I \times M}$ (words, words, news) where all co-occurrence entries are boolean and indicate whether the i^{th} and j^{th} words appear within the predefined window at least once. We then use CP/PARAFAC tensor decomposition to factorize the tensors. Another possibility is to create a text graph representation like in [14].
- **Use of pre-trained embedding** This simple method consists of doing the mean of the pre-trained embedding in each word like in [8, 9] to get the embedding of one article.
- **Latent Dirichlet Allocation** LDA is a generative probabilistic model for collections of discrete data such as text corpora. LDA is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is, in turn, modeled as an infinite mixture over an underlying set of topic probabilities. In the context of text modeling, the topic probabilities provide an explicit representation of a document [1].
- **Transformer for document embedding** Pre-training language model on a diverse corpus of unlabeled text [10] is a very effective approach on a wide range of benchmarks for natural language understanding. We used an open-source implementation of OpenAI's fine-tuned transformer language model in PyTorch.

4.1.2 k–nearest-neighbours

After transforming the article into a vector, we can then create our graph. For each node (article) we look for the k–nearest-neighbours (k–nearest articles) using a simple Euclidean distance in the embedding space. We also forced the graph to be undirected. Finally, we tunned the number k and it appears that for any value between 1 and 10, the results are equivalent.

4.2. Attention-based Graph Neural Network (AGNN)

The main focus of our work is the design of the graph using attention to classify articles [12]. First, we reproduced the results of the paper. Then, we applied AGNN on the Fake News Dataset and on the graph constructed in the previous section

4.2.1 Reproducing results on Benchmark datasets

We made two implementation of the paper, the first one with the library Pytorch-Geometric [3], and the second using only Pytorch based on the work of [7]. The benchmark datasets consist in citation network datasets. Documents are nodes and citation links are directed edges. Each node has a human annotated topic from a finite set of classes and a feature vector. We consider three datasets: Cora, Pubmed and Citesser. Description of these datsets are provided in Table 1. Although the networks are directed, we use undirected versions of the graphs for all experiments, as it is common in all baseline approaches. We use 16 units in the hidden layers and use 4 propagation layers for CiteSeer and Pubmed and 2 propagation layers for Cora. We row-normalize the input feature vectors, as it is standard in the literature. We used random weight initialization. Otherwise, we follow the instructions of the experiments provided in Appendix C in [12].

4.2.2 Application on the Fake News Graph

To test the Attention graph model, we will use the datasets presented in [6] which is comprised of 150 labelled articles: 75 are fake news articles and 75 are true. With this dataset, we will compare how the attention graph model performs and compare it with a baseline (fig. 1) using belief propagation to propagate the labels like in [5].

We use 16 hidden units, a learning rate of 0.01, a weight decay of 5e-4. We train our graph during 1000 epochs and we keep the one which has the best accuracy on the test set. We tunned the number of layers needed between 2 and 4.

5. Evaluation

5.1. Reproducing the results

We test on random splits of the same sizes. For 20 such randomly drawn dataset splits, average accuracy is shown in Table 2 with the standard error (the standard is present only with pytorch gemoetric implementation). We do not force equal number of labeled data for each class.

Our results are closed to the original results: we can apply the attention method on the FakeNews dataset.

5.2. Application on the Fake News Graph

We use the all public fake news dataset. We tested AGNN with many different scenarios. For every scenario, we used a percentage of labeled data with 2, 5, 10, $\cdots$ 50 every 5 percents then from 60 to 90 every 10% and 95%. We choose a number of neighbours between 1 and 5 and with 10. Moreover, we used a number of layer between 2 and 4. We repeated every scenario 20 times and we plotted the average and the standard deviation. We do not force our training subset to be balanced. That is to say, we do not force the labeled subset to have half of fake news data and half of non fake news.

All different scenarios are summarized in Table 3.

1. Co-occurence matrix scenarios with a Parafac rank decomposition of 10 and a window size of 5;
2. Mean of GloVe embeddings with an embedding of dimension 100;
3. Transformer embeddings of dimension 178.

Since we consider the fake article detection as a binary classification problem, we evaluated our method in terms of the following commonly used metrics:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

which is the percentage of correctly classified articles.

$$\text{Precision} = \frac{TP}{TP + FP}$$

which is the percentage of articles predicted as fake, out of all fake articles.

$$\text{Recall} = \frac{TP}{TP + FN}$$

which is the percentage of all fake articles that are correctly predicted as fake. and

$$\text{F1} = \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

which is the harmonic mean of precision and recall, and indicates a combined measure of performance. TP denotes true positives (correctly predicted fake news articles), FP denotes false positives (real news articles which were predicted as fake), TN denotes true negatives (i.e. correctly predicted real news articles), and FN denotes false negatives (i.e. fake news articles which were predicted as real). Results are similar for the two implementations (cf

Dataset	Nodes	Edges	Classes	Features	Labeled nodes
Cora	2,708	5,429	7	1,433	140
Pubmed	19,717	44,328	3	500	60
CiteSeer	3,327	4,732	6	3,703	120
Fake News	150	-	2	100	-

Table 1. Summary of the benchmark datasets as well as the Fake News dataset.

	Thekumparampil et. al [12]	Ours (based on [3])	Ours (based on [7])
Cora	81.0 ± 0.34	79.1	79.4
Pubmed	78.0 ± 0.46	78.4	-
CiteSeer	69.8 ± 0.35	68.20	69.3

Table 2. Classification accuracy evaluated with random splits of the data.

Figures 3 and 2) we will therefore keep the one from the method taken from [7].

We compare the results between the baseline and our method for 4 layers, 2 neighbours (best pipeline for [5]) and mean of GloVe embedding. Results are in Tables 4, 5 and 6. All the results are in Figures 2, 3, 4, 5, 6, 7, 8, 9.

- **Generalities:** First of all, [12] method is equivalent when we have less than 10% of labeled data. And as soon as we reach 10% this method outperforms [5] method. In fact, for 10% we reach 80% of accuracy whereas in the baseline it is around 70%. Moreover our results are more stable: the standard deviation is less important in our method.

- **Influence of the number of neighbours:** Compared with the baseline results where the number of neighbours is important in the quality of the result, no such deviation can be observed with the AGNN method. And this can be explained by the fact that we build the edges (k-nearest-neighbour graph) with the same information that we give to the nodes for classification. AGNNs consider both information, nodes and edges. So there is a redondency produces by this method of building the graph. To go further, 2 different methods of embedding to create the graph and to get the features could be used. Fig. 9 shows that the results are in fact different. To go further, it would be nice to use another methodology like in [14].

- **Influence of the Layers:** we observe that when we passed from 2 to 3 layers, results did not improve (see fig. 2 and 4). This is explained because the aim of the AGNN is to reduce the complexity of the network. But when we passed from 3 to 4, results seemed to be better (fig. 5).

- **Influence of the Embedding:** it seems that Glove and co-occurence matrix yield the best results.

5.3. Inter-class relevance score

Another way for analyzing the results and interpreting them would be to use the Inter-class relevance score. It is used to show the average attention from a node c_1 in one topic to another topic c_2 as presented in [12].

6. Conclusions

We develop a method which is stable, fast and which outperforms [5] by 10% when we only used 10% of labeled data. Further work is to extend this method to bigger and multi-label Fake News dataset. An other applications can also be sentiment analysis. It would be also good to compare this method with a GCN approach and its declination.

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[2] M. Conti, D. Lain, R. Lazzeretti, G. Lovisotto, and W. Quattrociocchi. It's always april fools' day!: On the difficulty of social network misinformation classification via propagation features. In *Information Forensics and Security (WIFS), 2017 IEEE Workshop on*, pages 1–6. IEEE, 2017.

[3] M. Fey, J. E. Lenssen, F. Weichert, and H. Müller. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[4] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, 2:729–734 vol. 2, 2005.

[5] G. B. Guacho, S. Abdali, N. Shah, and E. E. Papalexakis. Semi-supervised content-based detection of misinformation via tensor embeddings. *arXiv preprint arXiv:1804.09088*, 2018.

[6] B. D. Horne and S. Adali. This just in: fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. *arXiv preprint arXiv:1703.09398*, 2017.

Scenario Name	Node feature	Edge feature
Co-occurence matrix	Co-occurence matrix	Co-occurence matrix
GloVe	GloVe	GloVe
LDA	LDA	LDA
Transformer	Transfomer	Transfomer
Mix occ-Glove	Co-occurence matrix	GloVe
Mix Glove-occ	GloVe	Co-occurence matrix

Table 3. Different scenarios tested on our datasets

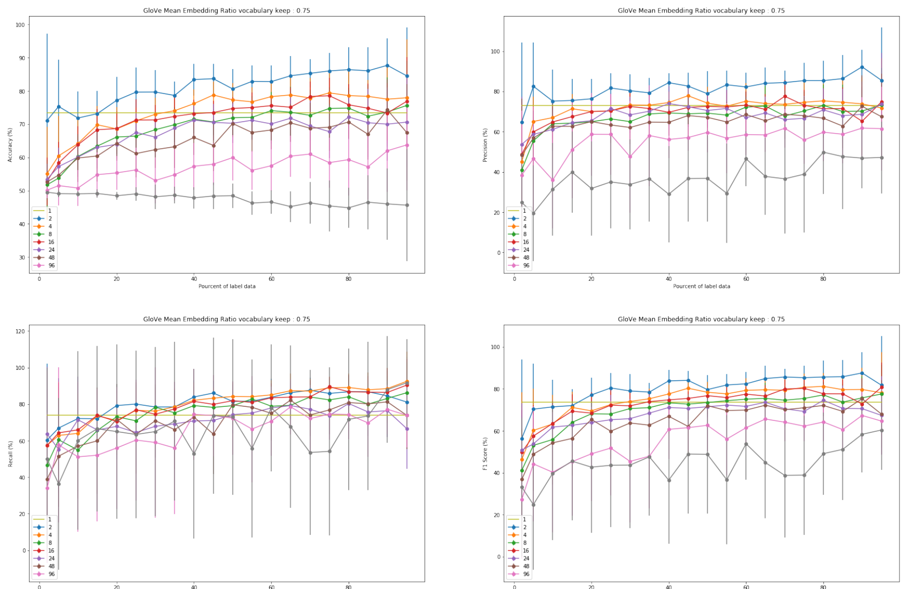

Figure 1. Baseline results from [5]. Each plot shows a metric with respect to the percentage of labeled data. The curve are obtained for different number of neighbours for the making of the graph. The yellow curve labeled "1" is the result obtained by [5] for 20% of labeled data.

[7] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[8] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[9] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[10] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_understanding_paper. pdf*, 2018.

[11] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

[12] K. K. Thekumparampil, C. Wang, S. Oh, and L.-J. Li. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735*, 2018.

[13] L. Wu and H. Liu. Tracing fake-news footprints: Characterizing social media messages by how they propagate. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 637–645. ACM, 2018.

[14] L. Yao, C. Mao, and Y. Luo. Graph convolutional networks for text classification. *arXiv preprint arXiv:1809.05679*, 2018.

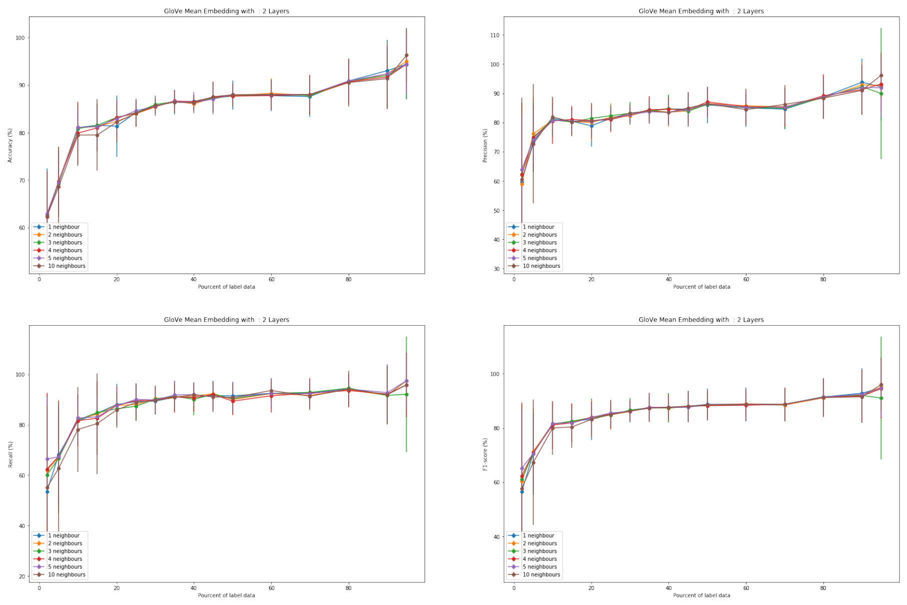

Figure 2. Results with GloVe embedding with AGNNs and 2 layers, implementation based on [7]

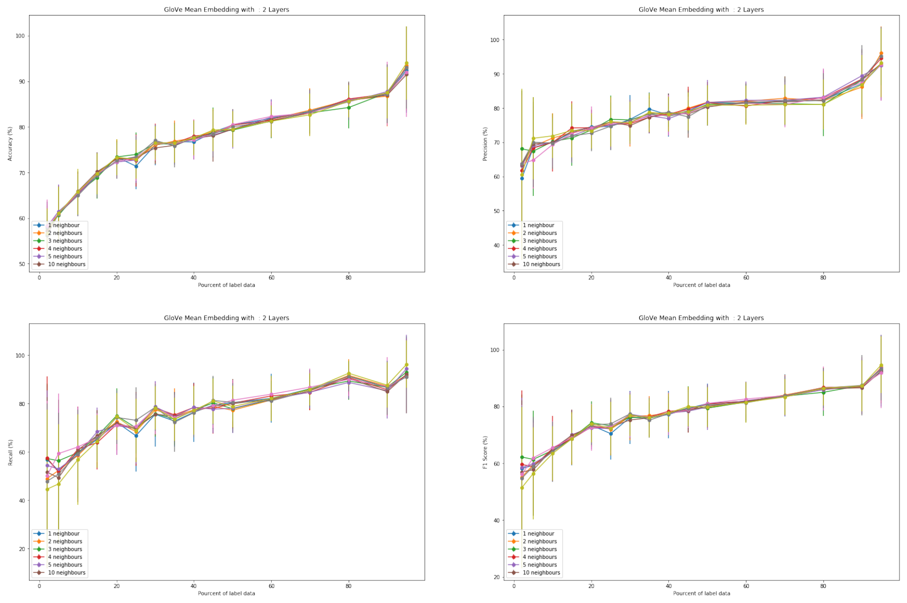

Figure 3. Results with GloVe embedding with AGNNs and 2 layers, implementation based on PyTorch geometric

Methods	**10%**			
	Accuracy	*Precision*	*Recall*	*F1-score*
Benchmark [5]	68.10 ± 7.4	68.80 ± 7.9	78.05 ± 15.6	71.6 ± 7.1
Our method (Glove 4 layers)	**78.11 ± 5.9**	**77.60 ± 4.9**	**79.83 ± 14.03**	**78.70 ± 7.3**

Table 4. Results with 10% of labeled data.

Methods	**15%**			
	Accuracy	*Precision*	*Recall*	*F1-score*
Benchmark [5]	73.41 ± 5.7	73.24 ± 6.7	78.86 ± 13.8	75.02 ± 7.2
Our method (Glove 4 layers)	**81.45 ± 5.9**	**80.47 ± 7.0**	**85.23 ±10.7**	**82.78 ± 8.4**

Table 5. Results with 15% of labeled data

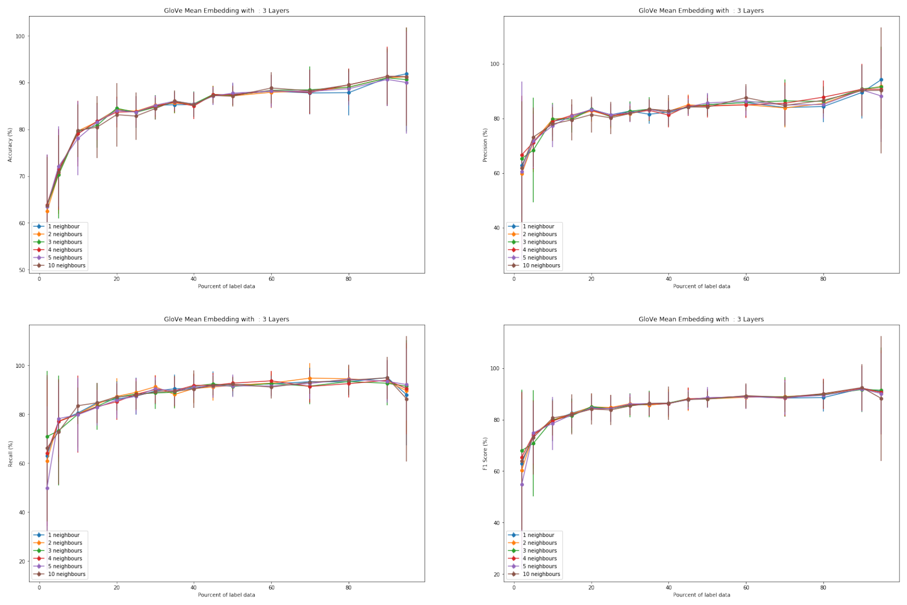

Figure 4. Results with GloVe embedding with AGNNs and 3 layers, implementation based on [7]

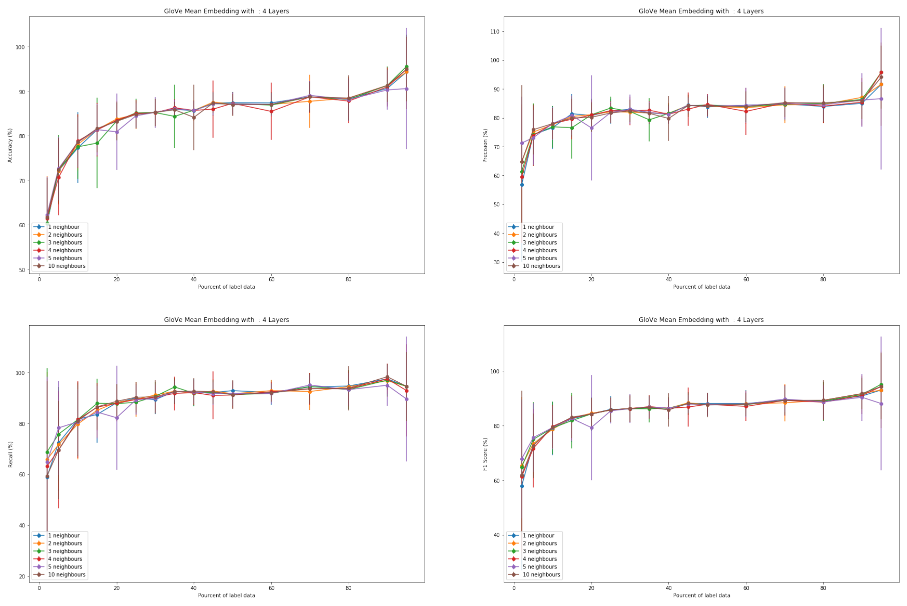

Figure 5. Results with GloVe embedding with AGNNs and 4 layers, implementation based on [7]

Methods	**20 %**			
	Accuracy	*Precision*	*Recall*	*F1-score*
Benchmark [5]	77.48 ± 4.7	74.15 ± 5.6	86.03± 12.1	78.9±5.3
Our method (Glove 4 layers)	**83.79 ± 4.1**	**81.05 ± 4.8**	**88.32 ±6.7**	**84.53 ± 5.6**

Table 6. Results with 20% of labeled data

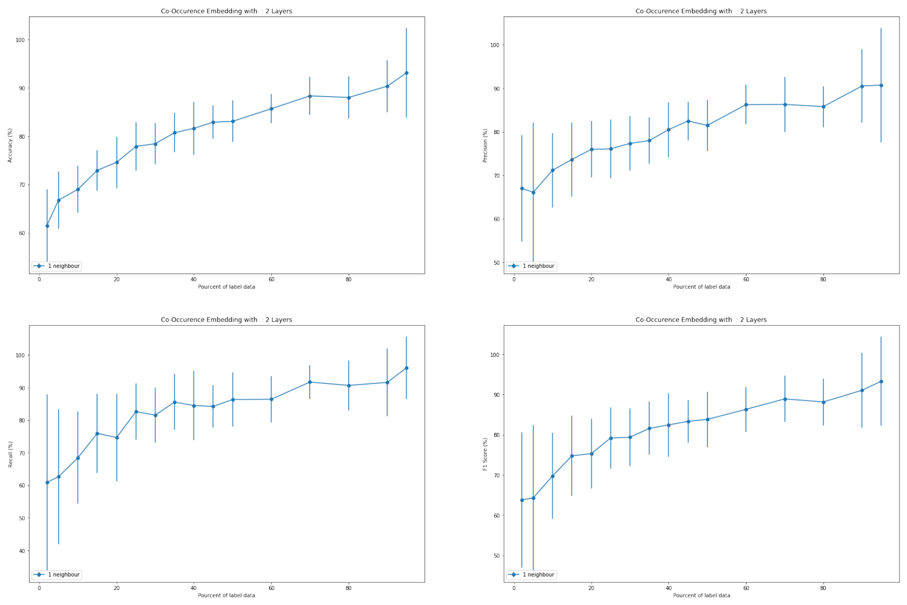

Figure 6. Results with AGNNs and co-occurence matrix embedding, 2 neighbours and 2 layers, implementation based on [7]

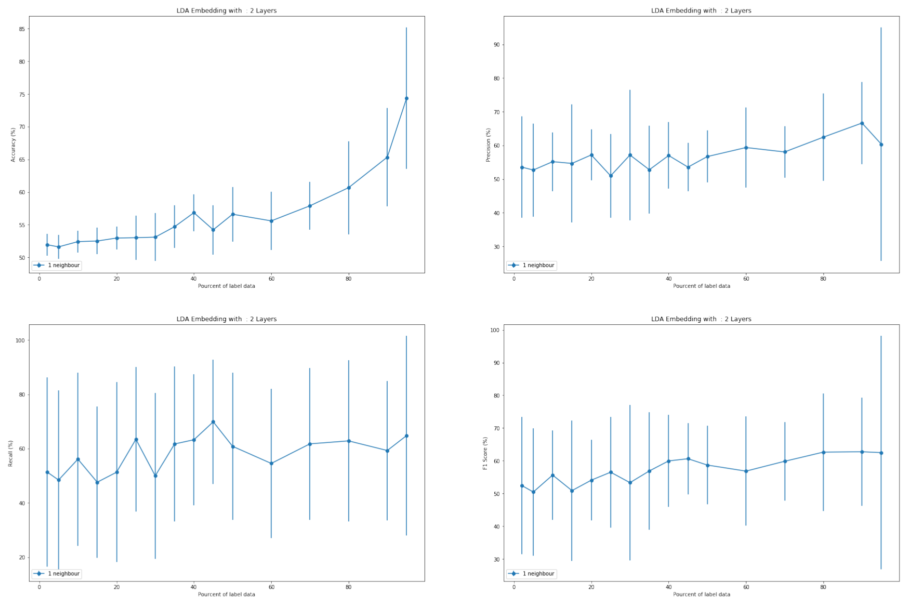

Figure 7. Results with AGNNs and LDA embedding, 2 neighbours and 2 layers, implementation based on [7]

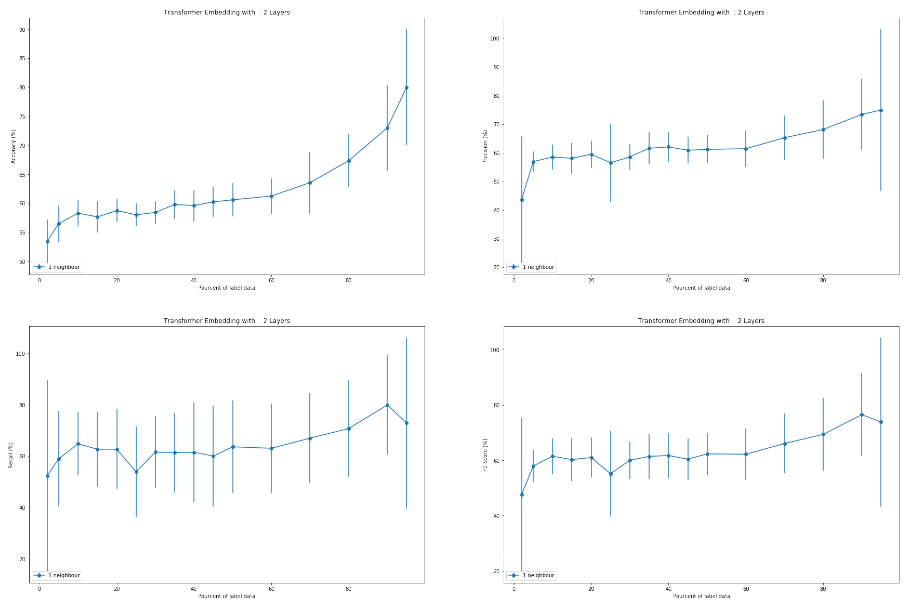

Figure 8. Results with AGNNs and Transformer embedding, 2 neighbours and 2 layers, implementation based on [7]

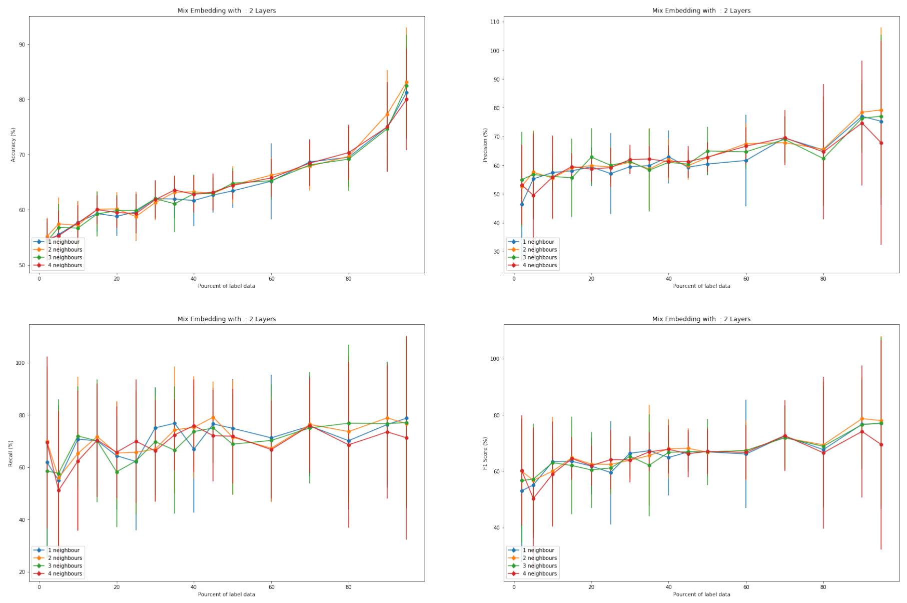

Figure 9. Results with AGGNs and Mix GloVe and co-occuence embedding, 2 neighbours and 2 layers, implementation based on [7]