HIGHER SCHOOL OF ECONOMICS

TERM PAPER

Extending NLP models via graph embeddings

Author:
Alexander ANDREEVSKIY

Supervisor:
Ilya MAKAROV

June 30, 2021

Acknowledgements

Model training, data preprocessing and embeddings construction pipeline is based on GraphText: Fusion of Graph and Text Information for ML Problems on Graphs.

Contents

Chapter 1

Introduction

With the rapid development of various machine learning pathways in recent years, one area that stands out is network representations. In general, graph is a natural way to represent information in a structured form. Many different scenarios can be modelled via networks: social connections and hierarchical structures, spread of information and propagation of influence, recommendation systems and classification problems.

Elaborating on the last one, graphs can be successfully applied for solving a classification problem for a set of objects (represented by nodes) and connections between them (edges). As for the document classification task, citations, attributions and any mentions in general can be viewed as edges, while titles of the paper (or other identifiers, i.e. authors) can serve as nodes. After the network structure has been constructed, nodes and edges are encoded through an embedding algorithm. Then the classifier predicts the node type.

It is important to mention that document classification task is also covered by the NLP methods. Document abstract or title, in some cases a full-length text, can be represented by a combination of vectors. In the most general way, words are mapped to vectors in a multidimensional space [Mikolov et al., 2013], usually Euclidean, although hyperbolic vector spaces also proved to be beneficial [Tifrea, Becigneul, and Ganea, 2019]. By design word similarity and context is preserved: words with similar meaning tend to have small cosine distance between them. Therefore, a specific document is related to a topic if the distance between the corresponding vectors is small.

Both graph and NLP approaches can be combined in a single framework - a fusion model, which preserves both topological structure and contextual information within text. Studies imply that such models can lead to an increased accuracy of classification [Ryabinin et al., 2020]. Therefore, their modification and further improvement is of high interest to researchers. This paper is dedicated to evaluating and describing the current state of fusion models along with best practices in the field.

Chapter 2

Related work

2.1 Text embeddings

In a document classification task text embeddings usually serve for representing the abstract of the paper or its' title. Both abstracts and titles often contain tag-words and key terminology, specific for the given field, thus these embeddings are helpful for identifying the type of the paper. As Harris, 1954 pointed out, "words that occur in the same context tend to have similar meanings", so that small distance may point out a semantic proximity.

A traditional way to construct text embeddings is through mapping words to a low-dimensional vector space, so that each word is represented by a vector [Mikolov et al., 2013; Bojanowski et al., 2017]. Then the document embedding is comprised of a combination of such embeddings. Therefore, similarity between documents can be measured by their distances.

A widely known model, Word2Vec, proposes two important additions to the general NLP framework: using skip-grams and continuous bag-of-words (CBOW). CBOW predicts the next word based on context, while the skip-gram predicts the context based on the word. This innovative approach helps to better capture semantic relationships between words.

As further research has shown, there is a lot of space for improving word embeddings. Such techniques as negative sampling, usage of pretrained embeddings, matrix factorization have helped to increase the variability of word representations. Moreover, concatenation of embeddings produced by different models often leads to better results [Wang et al, 2020], so models can be used all together.

Some of the models which are often referred to as benchmarks are listed below.

1. **Word2Vec**
   Word2Vec predicts a context from word (skip-gram) or a word from its context (continuous bag of words) [Mikolov et al., 2013].

2. **Doc2Vec**
   Doc2Vec extends Word2Vec approach to be able to learn continuous representations for texts of variable length (from a short sentence up to very long articles). Doc2Vec can preserve document context for very long sequences of words [Le and Mikolov, 2014].

3. **BERT**
   BERT is a multi-layer bidirectional transformer encoder, pretrained on a large corpus of text [Devlin et al., 2019]. It is designed to pretrain deep bidirectional representations from unlabeled text by conditioning on both left and right context. Embeddings depend on both the context of the sentence and the word

itself. Therefore, a single word can have multiple vector representations under different contexts.

4. **SBERT**
   SBERT uses Siamese network architecture to generate sentence embeddings [Reimers and Gurevych, 2019]. It adds pooling layer to the pretrained BERT model. There are 3 pooling strategies: using CLS token, taking MEAN or taking MAX of all output vectors. Afterwards softmax classifier is applied.

5. **BoW (bag-of-words) and TF-IDF (term frequency - inverse document frequency)**
   The classical methods of text representation. Each word is encoded as a one-hot vector, the documents is represented as a sum of representations of the words it contains.

2.2 Network embeddings

A classical way to represent graph is via adjacency matrix. However, this method yields very sparse matrices for large graphs and doesn't go beyond basic proximity measure. More advanced techniques include embeddings of edges and/or nodes via specific models.

Broadly speaking, there are 2 main methods of representation: based on a supervised learning (Skip-gram models, Laplacian eigenmaps, GNN) and based on unsupervised learning (DeepWalk, HOPE, Node2Vec).

Below is a brief description of models commonly used for network representation:

1. **DeepWalk**
   This model employs skip-gram approach on sampled random walks to learn embeddings [Perozzi, Al-Rfou, and Skiena, 2014]. DeepWalk uses simple unbiased random walks over the network. There's a parallel with text embeddings: the sequence of visited nodes during the random walk is treated as words of a sentence. Then DeepWalk trains neural networks by maximizing the probability of predicting context nodes for each target node in a graph.

2. **Node2Vec**
   This is a more efficient realization of random walk idea [Grover and Leskovec, 2016]. Unlike DeepWalk, Node2Vec employs a more flexible definition of random walks, with 2 hyperparameters, p and q, that represent bias. Through these parameters Node2Vec balances between BFS and DFS, which allows to preserve local and global graph structure.

3. **Graph2Vec**
   Graph2Vec employs a transductive approach to network embeddings. [Narayanan et al., 2017] The embedding is computed for a fixed set of graphs in a form of an embedding matrix (or look-up table) and so it only produces an embedding for graphs known at training time.

2.3 Fusion of text and graph embeddings

Combination of text and graph embeddings usually boosts the performance of classification models, since it takes more information into account. There are a number

of different options to combine embeddings. *The naive approach* is simple concatenation of text and network embeddings into one final embedding. It can be used as a baseline for comparison with the more advanced methods.

Graph Neural Networks (GCN, GAT, GraphSAGE, GraphGAN) can be trained on feature matrices constructed from BoW of TF-IDF matrices. Although such text embedding methods as Sent2Vec or Word2Vec might provide even better results.

1. **GCN**
   Graph Convolutional Network applies the adjacency matrix for filtering neighbourhood in the fully-connected layer [Kipf and Welling, 2016]. From the variety of existing architectures 4 were selected for the research: GAT (graph attention network) [Veličković et al., 2018], Chebyshev Spectral Graph [Defferrard, Bresson, and Vandergheynst, 2016], TAGCN [Du et al., 2018] and SGC [Wu et al., 2019]. GAT applies Graph Attention Network over an input signal. Chebyshev Spectral Graph uses Chebyshev filter is used with constant filter size k. TAGCN uses filters which are adaptive to the topology of the graph when they scan the graph to perform convolutionm with no approximation to the convolution is needed, thus TAGCN exhibits better performance than other GCNs. SGC is a model which reduces complexity through removing non-linearities and collapsing weight matrices between consecutive layers, saving on computation costs.

2. **TextGCN [Liang, Chengsheng, and Yuan, 2019], BertGCN [Lin et al., 2021], FastGCN [Chen, Ma, and Xiao, 2018]**
   TextGCN and BertGCN implement simultaneous, transductive learning of graph and text embeddings (parameters are learned from both train and test data). Both models add global semantic information on the corpus-level graph for better quality of embeddings. FastGCN also builds a global corpora, but utilizes inductive approach.

Chapter 3

Experimental design

3.1 Data preparation

3.1.1 Datasets

In our research we're going to perform the document classification on CORA dataset [Sen et al., 2008]. CORA is a citation network, in which each scientific paper is represented by node, while citations are represented as edges. There are 7 classes: Neural Networks, Reinforcement Learning, Probabilistic Methods, Rule Learning, Theory, Case-Based, Genetic Algorithms. In total network consists of 2 708 nodes and 5429 edges.

3.1.2 Data preprocessing

Before using any embeddings algorithm input texts are preprocessed. The pipeline is the following [adapted from Makarov, Makarov, and Kiselev, 2021]:

1. Converting all sentences to lowercase
2. Removal of stop words and special symbols
3. Removal of the most frequent words in the dataset, which appear in over 70% of the texts, and the rarest words, which appear less than 3 times
4. Lemmatization

3.1.3 Training and validation

Firstly, node representations are obtained using fusion approach (both text and graph embeddings). Next, the dataset is splitted into train and test subsets in different proportions (5%, 10%, 30% and 50% of labeled nodes). On the final step Logistic Regression classifier is trained.

3.2 Embeddings

3.2.1 Text Embeddings

Sets of hyperparameters for the models are taken from [Makarov, Makarov, and Kiselev, 2021]. Bag of Words and TF-IDF models use unigrams as input. BERT model is used with pretraining on English Wikipedia. BERT output has dim=160.

3.2.2 Network Embeddings

Only GCN-based network embedding methods were selected for the experiments: GAT, Chebyshev Spectral Graph (with filter size k=2, as in the original paper), TAGCN, SGC. Each model has 2 layers, adding more depth didn't improve the results.

3.3 Evaluation

The procedures described above are repeated two times on a set of different random seeds for different train/test proportions values. Then the mean and standard deviation of the results are reported for micro-F1 metric. One-Vs-Rest setting is applied, so that one classifier is trained for every class, while each model use samples from all other classes as "zero" class.

Chapter 4

Results

Fusion models were trained and tested on Cora dataset. The results of node classification are shown in the table below. Best scores are highlighted in bold.

TABLE 4.1: Fusion models on Cora (micro-F1 score).

% Labels	5%	10%	30%	50%
GAT + BERT	0.62 ± 0.01	0.70 ± 0.02	0.78 ± 0.00	0.79 ± 0.01
CHEB + BERT	0.60 ± 0.03	0.71 ± 0.02	0.80 ± 0.00	0.82 ± 0.01
TAG + TG-IDF	**0.80 ± 0.00**	**0.85 ± 0.00**	**0.87 ± 0.00**	0.87 ± 0.00
TAG + BOW	**0.80 ± 0.00**	0.84 ± 0.00	**0.87 ± 0.00**	0.87 ± 0.00
CHEB + BOW	0.79 ± 0.00	0.82 ± 0.00	0.86 ± 0.00	**0.88 ± 0.00**
CHEB + TF-IDF	0.79 ± 0.00	0.84 ± 0.00	0.86 ± 0.00	**0.88 ± 0.00**
SGC + BOW	0.78 ± 0.01	0.82 ± 0.00	0.86 ± 0.00	0.86 ± 0.00
SGC + TF-IDF	0.79 ± 0.00	0.83 ± 0.00	**0.87 ± 0.00**	**0.88 ± 0.00**
FastGCN	–	–	–	0.85 ± 0.00

Fusion models with TAG showed the highest f1-scores in all cases, except for 50% label ratio, with TG-IDF variation performing slightly better than with BoW. FastGCN, even though using global corpora, is placed below other models. Chebyshev Spectral Graph (denoted as Cheb) showed highest score with 50% label ratio, with both TG-IDF and BoW performing similarly well.

The results of classification with TAG and TF-IDF were visualised with t-SNE (dim=2) and are shown below.

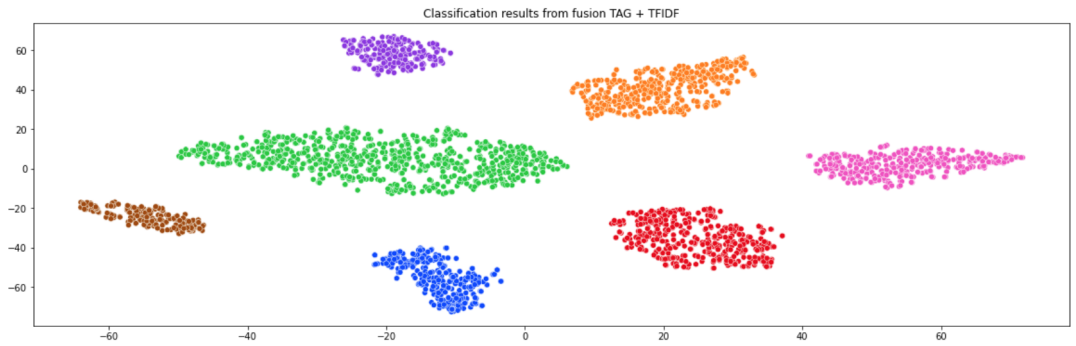

FIGURE 4.1: Fusion of TAG and TF-IDF

Chapter 5

Conclusion and future work

Experimental results indicate that fusion models, constructed from various architectures and layers of graph convolutional networks paired together with advance text embeddings methods lead to high performance on a document classification task. The optimal combination was TAG with TG-IDF, which managed to achieve up to 87% micro-f1 score on Cora dataset. Also, simultaneous learning of text and network embeddings, as in BertGCN and TextGCN, turned out to be very computationally expensive and overall didn't lead to an increase in quality, despite being finetuned on the dataset. Modification and improvement of current transductive models leaves a space for further research.

Bibliography

Bojanowski, P. et al. (2017). "Enriching Word Vectors with Subword Information". In: *Transactions of the Association for Computational Linguistics* 5, pp. 135–146.

Chen, J., T. Ma, and C. Xiao (2018). "FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=rytstxWAW.

Defferrard, M., X. Bresson, and P. Vandergheynst (2016). "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems*. URL: https://arxiv.org/abs/1606.09375.

Devlin, J. et al. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *NAACL-HLT*.

Du, Jian et al. (2018). *Topology Adaptive Graph Convolutional Networks*.

Grover, A. and J. Leskovec (2016). "node2vec: Scalable feature learning for networks". In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864.

Harris, Z. (1954). "Distributional structure". In: *Word* 10.2-3, pp. 146–162.

Kipf, T. and M. Welling (2016). "Semi-supervised classification with graph convolutional networks". In: URL: https://arxiv.org/abs/1609.02907.

Le, Q. V. and T. Mikolov (2014). "Distributed Representations of Sentences and Documents". In: *CoRR* abs/1405.4053. URL: http://arxiv.org/abs/1405.4053.

Liang, Y., M. Chengsheng, and L. Yuan (2019). "Graph convolutional networks for text classification". In: *In Proceedings of the AAAI Conference on Artificial Intelligence*.

Lin, Yuxiao et al. (2021). *BertGCN: Transductive Text Classification by Combining GCN and BERT*.

Makarov, I., M. Makarov, and D. Kiselev (2021). "Fusion of text and graph information for machine learning problems on networks". In: *PeerJ Computer Science* 7.

Mikolov, T. et al. (2013). "Efficient estimation of word representations in vector space". In: URL: https://arxiv.org/abs/1301.3781.

Narayanan, A. et al. (2017). "graph2vec: Learning Distributed Representations of Graphs". In: *CoRR* abs/1707.05005. URL: http://arxiv.org/abs/1707.05005.

Perozzi, B., R. Al-Rfou, and S. Skiena (2014). "DeepWalk: Online Learning of Social Representations". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710.

Reimers, N. and I. Gurevych (2019). "Sentence-BERT: sentence embeddings using Siamese BERT-networks". In: URL: https://arxiv.org/abs/1908.10084.

Ryabinin, M. et al. (2020). "Embedding Words in Non-Vector Space with Unsupervised Graph Learning". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sen, P. et al. (2008). "Collective Classification in Network Data". In: *AI Magazine* 29.3. DOI: 10.1609/aimag.v29i3.2157. URL: https://ojs.aaai.org/index.php/aimagazine/article/view/2157.

Tifrea, A., G. Becigneul, and O.-E. Ganea (2019). "Poincare glove: Hyperbolic word embeddings". In: *Proceedings of the 2019International Conference on Learning Representations*.

Veličković, P. et al. (2018). *Graph Attention Networks*.

Wu, Felix et al. (2019). *Simplifying Graph Convolutional Networks*.