Fusion of text and graph information for machine learning problems on graphs

Mikhail Makarov [1] , **Dmitrii Kiselev** [Corresp., 1] , **Ilya Makarov** [Corresp. 1, 2]

[1] HSE University, Moscow, Russia

[2] University of Ljubljana, Ljubljana, Slovenia

Corresponding Authors: Dmitrii Kiselev, Ilya Makarov
Email address: dkiseljov@hse.ru, iamakarov@hse.ru

Nowadays, an increased attention is drawn towards Network Representation Learning, which is a technique that maps nodes of a network into vectors of a low dimensional space, preserving their similarity. The vectors can later be used for different downstream tasks such as Node Classification, Link Prediction and Graph Visualization. Naturally, some networks have text information associated with them. For instance, in a citation network each node is a scientific paper associated with its abstract or title, in a social network all users might be viewed as nodes of a network and posts of each user as textual attributes. This work studies, how combination of existing methods can increase accuracy on the downstream tasks and proposes some modifications to popular architectures to better capture textual information.

Fusion of Text and Graph Information for Machine Learning Problems on Graphs

ABSTRACT

Nowadays, an increased attention is drawn towards Network Representation Learning, which is a technique that maps nodes of a network into vectors of a low dimensional space, preserving their similarity. The vectors can later be used for different downstream tasks such as Node Classification, Link Prediction and Graph Visualization. Naturally, some networks have text information associated with them. For instance, in a citation network each node is a scientific paper associated with its abstract or title, in a social network all users might be viewed as nodes of a network and posts of each user as textual attributes. This work studies, how combination of existing methods can increase accuracy on the downstream tasks and proposes some modifications to popular architectures to better capture textual information.

Keywords: graph embeddings, text embedding, information fusion, node classification, link prediction, node clustering, community detection, network science

1 INTRODUCTION

A lot of real-life problems can be modelled as graphs: citation networks, social networks, knowledge databases, etc. Ability to analyze such data structures is very important for the great variety of applications. For instance, social networks try to recommend for a user people to connect to, to do it one should solve a Link Prediction problem. Marketing departments of Telco companies might want to segment users according to their behaviour within a network of calls, which is the problem of Node Clustering. Biologists could need to find out the structural roles of proteins using their interaction network requiring a solution for Node Classification problem.

To be able to solve these problems, one has to devise the efficient representation of a network. Historically, the first way to represent a graph is the adjacency matrix. This representation has two major drawbacks: firstly, it captures only direct relationships between nodes, secondly, for real-life graphs, adjacency matrix tends to be very sparse and does not represent structural features apart from first-order proximity in a direct way.

Network Representation Learning (NRL) techniques were devised to mitigate the problems mentioned above. Its main idea to map nodes (or edges) of a graph into low dimensional space preserving their topological structure. The first NRL methods were mostly based on matrix factorization Roweis and Saul (2000) Belkin and Niyogi (2002). These methods do solve the dimensionality problem but are still very computationally expensive. More advanced approaches use random walks on graphs to approximate different kinds of similarity matrices Perozzi et al. (2014) Grover and Leskovec (2016). These methods are very scalable and therefore can be applied even for very big graphs.

Quite often nodes of a network have different kinds of attributes associated with them. This work is concerned with one type of attributes, such as text information. The problem of efficient representation of textual information is very similar to the same problem with graphs. The most classical techniques such as Bag of Words Harris (1954) and TF-IDF Salton and Buckley (1988) encode each word as one-hot vector and represent a document as a sum of representations of all words (using some coefficients). These methods are very simple but produce very sparse representations and do not take into account the order of words. More advanced approach, which is called Word2Vec Mikolov et al. (2013), employs a neural network (namely Skip-Gram model) to learn semantic of words, using their context. This method produces dense low dimensional embeddings, which makes it more powerful than the classic approaches. There are some extensions of Word2Vec Pagliardini et al. (2017) and Mikolov and Le (2014) that aim to learn document embeddings instead of embeddings for separate words. The most advanced models use bidirectional transformers Reimers and Gurevych (2019) to learn sensible embeddings.

The area of fusion of graph and text information for representation learning is not so well researched. The most simple approach is to learn network and document embeddings separately and then concatenate them to produce the final embedding. More sophisticated approaches include TADW Yang et al. (2015), which incorporates text attributes into matrix factorization problem, TriDnr Pan et al. (2016), which uses combined loss between Doc2Vec and DeepWalk algorithms, and GCN Kipf and Welling (2016), which uses special graph convolutional layers to employ attributes.

In this work the following contributions are made:

1. Different combinations of network and document embeddings are studied to improve the accuracy of the downstream tasks.

2. Some modifications are proposed to existing architectures to better take into account text information.

3. Comprehensive comparison of existing methods is performed.

2 RELATED WORK

In the real-life scenario, graphs quite often are accompanied by additional information. In this work, the main focus is put on one particular case, when each node of a network is associated with a text document.

2.1 Network Embeddings

There are a large variety of network embedding models for different cases. In the current work, we use in experiments only three models without node attributes.

2.1.1 DeepWalk

This model is used, because more advanced methods described below are based on it. DeepWalk (Perozzi et al. (2014)) simply samples random walks and learn to embed using a skip-gram approach similar to word2vec Mikolov et al. (2013).

2.1.2 Node2Vec

Node2Vec (Grover and Leskovec (2016))is more efficient realization of random walk idea. It balances between breadth-first and depth-first searches to keep trade-off between local and global network structure.

2.1.3 HOPE

HOPE (Ou et al. (2016)) is based on matrix factorization techniques. It is one of the best models in that group in terms of accuracy on different tasks. It preserves asymmetric transitivity by learning decomposition of corresponding graph distance measures like Katz index, Adamic-Adar or common neighbours.

2.2 Text embedding

2.2.1 Word2vec

The idea of it is to predict a context from word (skip-gram) or a word from its context (Continuous Bag of Word) Mikolov et al. (2013).

2.2.2 Sent2Vec

It is an extension of Word2Vec CBOW model, which was designed specifically to improve sentence embeddings. Firstly, it also learns embeddings for word n-grams. Secondly, it uses a whole sentence as a context window. Such an approach allow receiving better sentence embedding with n-gram aggregations.

2.2.3 Doc2Vec

Doc2Vec Mikolov and Le (2014) extends Word2Vec approach even further to be able to learn continuous representations for texts of variable length (starting from a short phrase to very long articles). Its main distinction from Sent2Vec is that Doc2Vec can preserve document context for very long sequences of words. Doc2Vec additionally create a lookup table with document embeddings. When target word is predicted this vector is concatenated to a source word vector.

2.2.4 SBERT

BERT Devlin et al. (2018) the main idea is to use bidirectional autoencoder, which means that it takes into account left and right contexts of each word. BERT model does not use any convolutional or recurrent layers instead if employs the attention mechanism to capture sequential information.

SBERT Reimers and Gurevych (2019) framework propose the idea to use Siamese network architecture to generate sentence embeddings. It adds pooling layer to the pre-trained BERT model. There are 3 pooling strategies: using CLS token, taking MEAN of all output vectors, taking MAX of all output vectors. Then softmax classifier is applied.

2.3 Naive mixture

The most straightforward method to fuse graph and text information is to first learn graph embeddings independently (giving up on text information) then learn text embeddings (again without taking into account the graph structure) and finally combine two types of embeddings by simply concatenating them. This method has the following advantages:

1. Graph and Document embeddings separately have been researched for quite a long time, so there are plenty of available methods/libraries etc.

2. Because embeddings for a network and texts are learned individually one have a lot of freedom so that one can choose a different dimension for text and graph embeddings, pre-train text embeddings on a completely different corpus of texts etc.

The main disadvantage is evident: text information is not taken into account while learning graph embeddings (and vice versa). It is important because two nodes might have the same distance in case of the graph proximity, but completely different semantic meaning.

2.4 Advanced mixture

TADW. One of the first attempts to actually incorporate text information into the process of network representation learning was made in TADW Yang et al. (2015) algorithm. The main idea was to enrich ordinary DeepWalk algorithm by taking into account text attributes. In order to do it the authors first prove that DeepWalk basically performs a matrix factorization process and extend it with TF-IDF feature matrix.

TriDNR. TriDNR tries to solve two issues of TADW: computational complexity of matrix factorization and missed word order in TF-IDF matrix encoding of texts. As the name suggests the algorithms learn the network representation using 3 (at max) sources:

1. Graph information.
2. Text information.
3. Label information.

In order to capture graph information DeepWalk algorithm is applied, for text and label information refined Doc2Vec is used.

Graph Convolutional Network. Kipf Kipf and Welling (2016) propose Graph Convolution Networks (GCN) as a light-weight approximation for the spectral convolution. This method provides better computational efficiency for semi-supervised tasks, like LPP or node classification Kipf and Welling (2016). One of the main advantages of GCNs is its ability to account node attributes. GCN simply applies the adjacency matrix for filtering neighbourhood in the fully-connected layer.

Generally, described fusion methods outperform document or network embeddings, but there is still some room for improvement: as for now, most researchers use Bag of Words or TF-IDF to produce input feature matrix for fusion methods such as TADW and GCN. It is promising to see how the combination of advanced document embeddings techniques with these methods might improve the accuracy of Machine Learning tasks (described in the next section). Also, GCN architecture might be improved so that word embeddings are learned simultaneously with network representations.

3 EXPERIMENTS

This chapter is devoted to series of experiments with the aim to find out which type of fusion of graph and text information can have an edge over graph or text information alone for Machine Learning problems on graphs.

3.1 Datasets

In order to be able to compare different kinds of algorithms described above chosen dataset should possess the following properties:

1. It should have a network structure, i.e. it should contain entities and relations between them.
2. At least some of the nodes should have text associated with it. It's important to note that texts associated with nodes should be in raw format (e.g. not in BOW). Although it is not required for every single node to have some text associated with it, still the more nodes have it, the better should be the quality.
3. At least some nodes should be associated with labels. This property is necessary to solve the node classification problem.

Cora Sen et al. (2008). Cora dataset is a citation network, where each node represent a scientific paper and each link show that one paper cites another one. There are 2708 nodes and 5429 edges in the network. Each node has text with a short description (abstract) of the paper. Average text length in words is 130. All nodes are grouped into 7 classes: Neural Networks, Rule Learning, Reinforcement Learning, Probabilistic Methods, Theory, Case-Based, Genetic Algorithms. The network does not contain any isolated nodes.

CiteSeer-M10 Lim and Buntine (2016). This dataset is a subset of original CiteSeer data, which contains scientific publications in different disciplines grouped into 10 different classes. M10 version consists of 38 996 nodes and 76 630 edges. However, only 10 310 nodes have the text (paper title) and label information associated with them. Average text length in words is 9. In this case, text information contains only the name of the paper (rather than the abstract). Some of the nodes are isolated, which makes this dataset generally harder than the previous one.

DBLP Yang and Leskovec (2015). DBLP is a bibliographic system for computer science publications. Publications (described by the title) might be connected to each other via citation. In this work only subset Pan et al. (2016) of the network is used, containing 60 744 nodes (all accompanied with text and label attributes) and 52 890 edges. Average text length in words is 8.

3.2 Document Embeddings

The first part of the experiments is mainly concerned with estimating whether textual information alone is sufficient to effectively solve Machine Learning Problems on text networks. Intuitively, in case of citation networks text description should be correlated with the target class (the topic of research), so results on text data might be a good baseline using which other types of embeddings are compared. Questions to be addressed in this section:

1. Do advanced document embedding techniques (Sent2Vec, Doc2Vec, SBERT) generally outperform classic approaches (such as BOW, TFIDF) in case of citation networks?
2. How a share of train data (in comparison with test) affects the prediction power of the model?
3. How average text length influences model quality?
4. Whether models pre-trained on huge amount of data perform better than models that are trained “from scratch”?

One of the most crucial steps to deal with this problem is text preprocessing. Preprocessing is performed before embedding algorithm is applied and includes the following procedures:

3.2.1 Normalization

All non-alphanumeric symbols (punctuation, tags, etc.) are removed, the text is lowercased, all extra spaces are removed, words that are less than 3 symbols are removed.

3.2.2 Stop words removal

Stop words are the subset of the words that are used frequently in a particular language and do not carry any specific meaning. For instance, in case of English the following words might be considered as stop words: ”also”, ”and”, ”any”, ”be”, etc.

3.2.3 Lemmatization

During this step each word is converted to its initial form. For instance ”better” is converted to ”good”. By doing so a downstream model can better generalize.

3.2.4 Filtering

Words that are very rare (appear less than 3 times) or very frequent (appear in at least 70% of all documents) are removed.

Bag of Words and TF-IDF models use only unigrams as input since datasets are relatively small and choosing higher ngram_range will lead to poor generalization. For LDA algorithm gensim [1] implementation was used because it can use multiple cores to fit and therefore show much better performance. The following hyperparameters are used for LDA: number of topics (effectively embedding size) = 20, $\alpha = 0.1$, $\beta = 0.1$. For Word2Vec, Sent2Vec and Doc2Vec there are two different settings: using a pre-trained model and train a model from scratch. When trained from scratch every model has two variations: with embedding, dimension equals to d and 64, where d is a dimension for pre-trained model and is always greater than 64. The intuition for this is the following: when the amount of data is not very big and the dimension of embedding is a quite high model might overfit easily, in the most extreme case model might just learn one-hot encoding for every word. Therefore it is sensible to try to lower dimension size (64 in this case). However this is not the case for pre-trained models, because they harness the huge amount of data and therefore can afford to choose bigger d (around $100-600$), still, it is important to train a model from scratch using d, which equals the embedding dimension for the pre-trained version,

[1] https://radimrehurek.com/gensim/models/ldamodel.html

to be able to better compare the results. Word2Vec was pre-trained on Google-News dataset Mikolov et al. (2013), which contains around 100 billion words, vector size for the pre-trained model is 300. Scratch model (gensim implementation [2]) is trained using the following parameters: vectors size = 300 (or 64), window size is equal to 5, $\alpha = 0.025$, ns_exponent parameter equals to 0.75. Doc2Vec (also implemented in gensim [3]) was pre-trained Lau and Baldwin (2016) using English Wikipedia dataset. Scratch version was trained using Distributed Memory mode with vector size = 300 (or 64). Sent2Vec [4] was also pre-trained Pagliardini et al. (2018) using English Wikipedia with vector size = 600. Scratch model was trained for 100 epoch with learning rate 0.05 using negative sampling. Used SBERT was pre-trained [5] on SNLI dataset Bowman et al. (2015), which consists of 570 000 sentence pairs divided into 3 classes: contradiction, entailment and neutral. The reason is that to train SBERT from scratch or at least to fine-tune it on the custom data, one needs to feed pair of documents as input, where each pair is similar or dissimilar, this is not directly applicable to the citation networks data format considered here.

3.3 Network Embeddings

In citation networks, papers from one field tend to cite each other much more frequently than papers from other fields therefore network structure should give very important insights for the node classification and link prediction tasks. Another important issue is to compare how well network embeddings perform in comparison with document embeddings on different datasets.

For the experiments with poor network embeddings, three network embeddings were selected: HOPE, Node2Vec and DeepWalk. The reason for such a choice is quite straightforward: these methods tend to outperform others in most settings. For HOPE implementation from GEM [6] library is used. Hyperparameter β is chosen to be 0.01 (as it is used in other papers). For DeepWalk original implementation [7] is used, with the following hyperparameters: vector size - 10, number of walks per-vertex - 80, window size - 10. Node2Vec also follows original implementation [8] with the following hyperparameters: vector size - 10, number of walks per-vertex - 80, window size - 10 (the same as DeepWalk).

3.4 Fusion of text and graphs

3.4.1 Naive Combination

Document embeddings and network embeddings are learned separately, for every node final embedding is represented as concatenation for the corresponding document embedding and network embedding. This method can be viewed as a good baseline for fusion methods. Bag of Words and Sent2Vec are used for document embeddings and DeepWalk for network embeddings.

3.4.2 TADW

Two versions of TADW are used: with TF-IDF and Sent2Vec for the feature generation. The following hyperparameters are used: vector size = 160, number of iterations = 20, $\lambda = 0.2$. SVD is used on input feature matrix to reduce its dimension to 200 (as in the original paper).

3.4.3 Tri-DNR

All three sources are used: texts, network and labels to get the final embeddings. Clearly, only labels from the train set are present while others are masked. The following hyperparameters are used: vector size = 160 (to match TADW), text weight = 0.8, passes = 50.

3.4.4 GCN

in most of the papers authors use simple BOW or TF-IDF matrices as a feature matrix for GCN. It might be sensible to experiment with more advanced document embeddings techniques to improve the results as we have already seen that for some settings Sent2Vec or Word2Vec outperform BOW and TF-IDF. The model is trained for 200 epochs using Adam optimizer, the best model (according to validation results) is saved. The vector size is equal to 64, the model contains two convolutional layers.

[2]https://radimrehurek.com/gensim/models/word2vec.html
[3]https://radimrehurek.com/gensim/models/doc2vec.html
[4]https://github.com/epfml/sent2vec
[5]https://github.com/UKPLab/sentence-transformers
[6]https://github.com/palash1992/GEM
[7]https://github.com/phanein/deepwalk
[8]https://github.com/aditya-grover/node2vec

Also it interesting to try new modification for GCN architecture: instead of using a fixed feature matrix as input, one can replace it with a lookup table with learnable embeddings. This way model can simultaneously learn text embeddings as well as network embeddings. In this case padded (to ensure fixed length) sentences of tokens are fed as input, then lookup table with embeddings is used. After that in order to obtain embeddings for sentences mean and max functions applied for word embeddings of each sentence. Then the rest of the network is the same as for ordinary GCN.

3.5 Validation

3.5.1 Node classification

firstly, input data (text, graph or both) is preprocessed, then nodes of the network are mapped into latent space using the input information. After that data is split into train/test subsets using different train ratios (5%, 10%, 30% and 50%). Then Logistic Regression classifier is fit on train data. Finally, the result is evaluated on the test set.

3.5.2 Link prediction

first of all, edges of the graph are randomly split into train test with the specified train ratio. Then test edges are masked (effectively removed) from the graph. Then for every edge in the train set another non-existing edge is added. Existing edges are marked as ”ones” and non existing as ”zeros”. The same is done for the test set also. Then the modified graph is used to learn node embeddings (using text or graph information or both). After embeddings for all of the nodes are learned Hadamard product $x_u \odot x_v \equiv x_{u,i} * x_{v,i}$ is applied to get embeddings for edges in both train and test sets. After that Logistic Regression classifier is trained using a train set with the target variable indicating the presence or absence of edge. Then quality is validated on the test set.

For both cases, this procedure is repeated 5 times with random train/test splits for different values of train ratio. The mean and standard deviation of the results is reported. As a quality metric F_1 score is used.

For Logistic Regression sklearn [9] implementation (in python) is used with lbfgs solver, L_2 penalty and $C = 1$. For multiclass classification (number of classes ¿ 2) OneVsRest Brownlee (2020) setting is applied, which means that one classifier is trained for every class, besides each model use samples from all other classes as ”zero” class.

4 RESULTS

4.1 Node classification

Table 1 shows the comparison between classic text approaches on Cora dataset. First of all, these techniques show very decent metrics, especially when the percentage of labelled nodes is not very small. The best algorithm is Bag of Words, which outperforms every other classic method, it also shows quite a good stability even for a very small per cent of labelled nodes in the training sample. TF-IDF performs similarly on 30% and 50% of labelled nodes but degrades significantly on the lower values. Although LDA results are not very high, it shows a very consistent result across different shares of labelled nodes.

% Labels	5%	10%	30%	50%
BoW	**0.63±0.01**	**0.68±0.01**	**0.76±0.01)**	**0.78±0.01**
TF-IDF	0.35±0.01	0.49±0.01	0.70±0.01	0.76±0.01
LDA	0.49±0.01	0.57±0.01	0.60±0.01	0.61±0.01

Table 1. Classic Document Embeddings on Cora (micro-F_1)

Table 2 presents results for more advanced methods on Cora dataset. Models trained from scratch, despite being significantly simpler and easier to train, generally performed better than their pre-trained counterparts. The reason for that is probably the fact, that language, which is used to describe the scientific paper in computer science, significantly differs from the language used in news or (average) Wikipedia article. The by far best model out of advanced document embeddings is Sent2Vec (trained from scratch), it also shows very consistent results concerning the different share of labelled nodes. Another insight is that advanced embeddings could not beat the Bag of Words technique when the share of labelled data is

[9] https://scikit-learn.org/stable/modules/generated/

high enough. This might be explained by the nature of the data. Abstract for scientific papers is just a set of keywords, in this case, the Bag of Words hypothesis is applied very well. However, when there is a small percentage of labelled data (which is the practice case), advanced embeddings significantly outperform Bag of Words, which means that they tend to generalize better. One can note that choosing different values for embeddings dimension does not significantly influence the results.

% Labels	5%	10%	30%	50%
SBERT pretrained	0.57±0.01	0.61±0.01	0.68±0.01	0.70±0.01
Word2Vec pretrained	0.34±0.01	0.44±0.01	0.59±0.01	0.63±0.01
Word2Vec (d=300)	0.64±0.01	0.68±0.01	0.70±0.01	0.71±0.01
Word2Vec (d=64)	0.65±0.01	0.68±0.01	0.70±0.01	0.72±0.01
Doc2Vec pretrained	0.54±0.01	0.61±0.00	0.65±0.01	0.67±0.01
Doc2Vec (d=300)	0.49±0.01	0.58±0.01	0.66±0.01	0.68±0.01
Doc2Vec (d=64)	0.50±0.02	0.58±0.01	0.65±0.00	0.67±0.01
Sent2Vec pretrained	0.63±0.02	0.69±0.01	0.74±0.01	**0.77±0.01**
Sent2Vec (d=600)	**0.68±0.02**	**0.72±0.01**	**0.75±0.01**	**0.77±0.01**
Sent2Vec (d=64)	**0.68±0.02**	**0.72±0.01**	**0.75±0.01**	**0.77±0.01**

Table 2. Advanced Document Embeddings on Cora (micro-F_1)

Table 3 and Table 4 present the results on CiteSeer-M10 dataset. This dataset differs from the first one in a sense that texts are significantly shorter, but the total amount of nodes is bigger. One can note that although Bag of Words is still the best out of the classic techniques, on CiteSeer-M10 TF-IDF performs almost as good. The performance does not degrade so significantly when the percentage of labelled nodes becomes small. The reason for that is twofold: firstly, the absolute number of samples is bigger, so it is easier to generalize, secondly, text length is much smaller, therefore, there is less ”variation” in the data. Predictably, LDA performed even poorer than on the first dataset. It is a known issue that LDA does not generally perform well on small datasets because it is much harder to extract ”topics” from a few words.

% Labels	5%	10%	30%	50%
BoW	**0.62±0.00**	**0.66±0.00**	**0.73±0.01**	**0.76±0.01**
TF-IDF	0.61 ±0.01	**0.66±0.01**	0.72±0.01	0.75±0.00
LDA	0.37±0.01	0.38±0.00	0.39±0.00	0.39±0.00

Table 3. Classic Document Embeddings on Citeseer-M10 (micro-F_1)

Considering advanced document embedding techniques (Table 4), all architectures (at least in one of the configurations) outperforms classic methods, when the percentage of labelled nodes is small (5% or 10%). When the share of labelled data is bigger they show performance similar to Bag of Words. Opposite to the Cora results, here one can note that pre-trained version of all models substantially outperforms their counterparts trained from scratch. The explanation for that might be the fact that text length is quite short and the amount of data is not enough to restore dependencies between words.

Table 5 presents the results of classic document embeddings methods on DBLP dataset. One can that the results are quite similar to the ones achieved on CiteSeer-M10 dataset: Bag of Words and TF-IDF performs equally good with the former performing slightly better. Also, there is no dramatic degradation in the score when the percentage of labelled nodes is small.

Regarding advanced methods on DBLP dataset Table 6 one can see that Sent2Vec outperforms all other architectures. Word2Vec also show very decent results (especially in terms of stability). For Doc2Vec (in opposite to Sent2Vec) pre-trained version performed far better than the one trained from scratch. Again advanced embeddings outperform classic techniques when it comes to a small percentage of labelled data and performs almost as good in case of more labelled data.

To sum up document embeddings experiments:

1. Advanced document embeddings techniques such as Sent2Vec, Doc2Vec, Word2Vec outperform classic approaches such as Bag of Words and TF-IDF when the percentage of labelled data is small (< 30%) and performs similarly when it is higher. LDA generally shows bad performance on all

% Labels	5%	10%	30%	50%
SBERT pretrained	0.66±0.00	0.68±0.00	0.72±0.01	0.73±0.01
Word2Vec pretrained	0.67±0.00	0.69±0.00	0.72±0.00	0.73±0.01
Word2Vec (d=300)	0.55±0.00	0.57±0.00	0.59±0.00	0.60±0.01
Word2Vec (d=64)	0.58±0.00	0.59±0.00	0.61±0.00	0.62±0.01
Doc2Vec pretrained	**0.68±0.00**	**0.70±0.00**	**0.74±0.01**	**0.75±0.00**
Doc2Vec (d=300)	0.53±0.00	0.56±0.00	0.59±0.00	0.61±0.00
Doc2Vec (d=64)	0.56±0.01	0.59±0.00	0.62±0.00	0.63±0.00
Sent2Vec pretrained	**0.68±0.00**	**0.70±0.00**	0.73±0.01	**0.75±0.01**
Sent2Vec (d=600)	0.64±0.01	0.66±0.00	0.70±0.01	0.71±0.01
Sent2Vec (d=64)	0.63±0.01	0.65±0.00	0.68±0.00	0.69±0.01

Table 4. Advanced Document Embeddings on Citeseer-M10 (micro-F_1)

% Labels	5%	10%	30%	50%
BoW	**0.75±0.00**	**0.77±0.00**	**0.79±0.00**	**0.80±0.00**
TF-IDF	0.74 ±0.01	0.76±0.01	**0.79±0.01**	**0.80±0.00**
LDA	0.54±0.00	0.55±0.00	0.55±0.00	0.56±0.00

Table 5. Classic Document Embeddings on DBLP (micro-F_1)

three datasets. Although pre-trained SBERT model shows decent results on CiteSeer-M10 and DBLP, still it was outperformed by other architectures and even classic approaches.

2. In general, advanced embeddings techniques and LDA show very consistent results even for a small percentage of trained labels, whereas Bag of Words and TF-IDF show degrading results when there is a small share of labelled nodes. However when text information is short (only titles) and there is more (in absolute values) trained data this effect is mitigated.

3. TF-IDF and Bag of Words generally performs better for short text (paper titles), because they are basically set of keywords. Advanced methods show good performance in both settings (short and long texts).

4. One can see that in some cases pre-trained models perform better and in some cases worse, so it is better to experiment with both of the approaches.

% Labels	5%	10%	30%	50%
SBERT pretrained	0.69±0.00	0.72±0.00	0.75±0.01	0.75±0.01
Word2Vec pretrained	0.72±0.01	0.73±0.01	0.74±0.00	0.74±0.01
Word2Vec (d=300)	0.76±0.00	0.76±0.00	0.77±0.00	0.77±0.01
Word2Vec (d=64)	0.76±0.01	0.76±0.00	0.76±0.00	0.77±0.00
Doc2Vec pretrained	0.73±0.00	0.75±0.00	0.76±0.00	0.76±0.00
Doc2Vec (d=300)	0.55±0.01	0.56±0.00	0.57±0.00	0.58±0.00
Doc2Vec (d=64)	0.54±0.01	0.54±0.00	0.55±0.00	0.55±0.00
Sent2Vec pretrained	0.73±0.00	0.75±0.00	0.77±0.01	**0.77±0.01**
Sent2Vec (d=600)	**0.77±0.00**	**0.78±0.00**	**0.79±0.00**	**0.79±0.01**
Sent2Vec (d=64)	**0.77±0.01**	**0.78±0.00**	0.78±0.00	0.78±0.00

Table 6. Advanced Document Embeddings on DBLP (micro-F_1)

According to the results on Cora dataset (Table 7) DeepWalk and Node2Vec show similar performance with DeepWalk being slightly better when the percentage of labelled nodes is bigger than 5%. HOPE shows very poor results (near to random) for Node Classification task. Comparing the results with document embeddings techniques (Tables 1 and 2) one can note that DeepWalk and Node2Vec outperform

% Labels	5%	10%	30%	50%
DeepWalk	0.72±0.01	**0.77±0.00**	**0.81±0.00**	**0.82±0.01**
Node2Vec	**0.74 ±0.01**	0.76±0.01	**0.80±0.00**	**0.81±0.01**
HOPE	0.29±0.00	0.30±0.00	0.30±0.00	0.31±0.00

Table 7. Network Embeddings on Cora (micro-F_1)

all other algorithms by a significant margin. Moreover, the tendency holds for different values of labelled nodes. It generally means that for Cora network data has a higher correlation with the target.

For Citeseer-M10 dataset (Table 8) DeepWalk and Node2Vec show identical performance for all values of labelled nodes, whereas HOPE again performs quite poorly. Interestingly, in contrast with Cora, here one can see that document embeddings techniques outperform network embeddings.

% Labels	5%	10%	30%	50%
DeepWalk	**0.63±0.00**	**0.65±0.01**	**0.67±0.00**	**0.68±0.00**
Node2Vec	**0.63±0.01**	**0.65±0.00**	**0.67±0.00**	**0.68±0.00**
HOPE	0.12±0.00	0.13±0.00	0.17±0.00	0.20±0.00

Table 8. Network Embeddings on Citeseer-M10 (micro-F_1)

Table 9 shows the results for network embeddings methods on DBLP dataset. Similar to the Citeseer-M10 dataset here we can see that DeepWalk and Node2Vec perform equally. Also one can see that document embeddings techniques severely outperform network embeddings on this dataset.

Generally, different datasets show different importances for text and network data. For some datasets nodes from the same class tend to link each other (the phenomenon is called homophily Barabási and Pósfai (2016)), which means that network structure is really useful for predicting the target. For other datasets nodes might as well tend to cite nodes from other classes, in this case, network information is less useful. Even though on some datasets one type of information (text or network) significantly outperforms the other, still both might be useful as they tend to provide complementary information (different ”views” on target).

% Labels	5%	10%	30%	50%
DeepWalk	**0.52±0.00**	**0.53±0.00**	**0.53±0.00**	**0.53±0.00**
Node2Vec	**0.52±0.00**	**0.53±0.00**	**0.53±0.00**	**0.53±0.00**
HOPE	0.29±0.01	0.30±0.01	0.31±0.00	0.31±0.00

Table 9. Network Embeddings on DBLP (micro-F_1)

Analyzing the results on Cora dataset (Table 10) one can note that a naive combination of textual and network features performs almost as good as more advanced approaches such as TADW and TriDNR. However, GCN significantly outperforms all other approaches. Also, all approaches except TriDNR perform better than methods that use only text or only network information, so one can conclude that these two types of information are complementary and should be considered.

% Labels	5%	10%	30%	50%
BOW + DeepWalk	0.74±0.01	0.80±0.01	0.84±0.00	0.86±0.01
Sent2Vec + DeepWalk	0.76±0.01	0.79±0.00	0.84±0.01	0.85±0.01
TADW - TF-IDF	0.72±0.02	0.80±0.01	0.85±0.01	0.86±0.01
TADW - Sent2Vec	0.75±0.01	0.80±0.01	0.83±0.00	0.85±0.00
TriDNR	0.59±0.01	0.68±0.00	0.75±0.01	0.78±0.01
GCN - TF-IDF	**0.80±0.01**	**0.83±0.01**	**0.86±0.01**	**0.87±0.01**
GCN - Sent2Vec	0.77±0.01	0.82±0.00	0.85±0.01	**0.87±0.01**

Table 10. Fusion methods on Cora (micro-F_1)

Tables 11 and Table 12 show that unlike Cora dataset naive combination of BOW and DeepWalk significantly outperforms much more advanced algorithms. GCN modifications show the best results on all percentages of training nodes except for 5%. Also, the fusion of text and graph information shows superior results to network or text embeddings alone.

% Labels	5%	10%	30%	50%
BOW + DeepWalk	**0.73±0.01**	0.76±0.00	0.81±0.01	0.83±0.01
Sent2Vec + DeepWalk	0.73±0.01	0.75±0.00	0.79±0.01	0.80±0.01
TADW - TF-IDF	0.47±0.02	0.51±0.01	0.57±0.01	0.59±0.01
TADW - Sent2Vec	0.57±0.01	0.60±0.00	0.65±0.01	0.66±0.01
TriDNR	0.63±0.01	0.68±0.00	0.74±0.01	0.77±0.01
GCN - TF-IDF	0.71±0.01	0.76±0.01	0.81±0.01	0.83±0.01
GCN - Sent2Vec	**0.73±0.01**	**0.80±0.00**	**0.84±0.01**	**0.87±0.01**

Table 11. Fusion methods on Citeseer-M10 (micro-F_1)

What is more, it is quite interesting to note that fusion of Sent2Vec + GCN show very stable results across different ratio of labelled nodes and outperforms TF-IDF + GCN when it comes to a very small training sample.

% Labels	5%	10%	30%	50%
BOW + DeepWalk	0.77±0.02	0.79±0.01	**0.81±0.01**	0.82±0.01
Sent2Vec + DeepWalk	**0.78±0.01**	**0.80±0.00**	0.80±0.01	0.80±0.01
TriDNR	0.72±0.01	0.75±0.00	0.78±0.01	0.79±0.01
GCN - TF-IDF	0.71±0.01	0.76±0.01	**0.81±0.01**	**0.83±0.01**
GCN - Sent2Vec	**0.78±0.01**	**0.80±0.00**	**0.81±0.01**	0.81±0.01

Table 12. Fusion methods on DBLP (micro-F_1)

4.2 Link prediction

For the task of link prediction, one can expect document embeddings to perform more consistently concerning train ratio in comparison with network embeddings. Because network embeddings techniques ”suffer” twice when the percentage of train data is decreased: firstly, it affects initial graph so it is harder to learn embeddings itself, secondly, it is harder to train a classifier using less data, whereas document embeddings have only the second problem is they are not dependent on the graph structure.

Table 13 shows results of document embeddings techniques on Cora dataset. Again one can see that BOW outperforms other methods, but for link prediction (contrary to Node Classification) LDA demonstrates much better performance. Similar to Node Classification problem on Link Prediction advanced network embeddings perform worse when the percentage of train data is high but show better results when it gets lower.

% Train edges	5%	10%	30%	50%
Bag of Words	0.69±0.01	0.71±0.00	**0.75±0.01**	**0.76±0.00**
LDA	0.68±0.01	0.69±0.01	0.71±0.01	071±0.01
SBERT pretrained	0.69±0.00	0.71±0.00	0.74±0.01	**0.76±0.01**
Word2Vec (d=300)	0.68±0.00	0.70±0.00	0.72±0.00	0.73±0.01
Doc2Vec (d=300)	0.67±0.01	0.70±0.00	0.73±0.00	0.74±0.00
Sent2Vec (d=600)	**0.71±0.00**	**0.72±0.01**	**0.75±0.00**	**0.76±0.01**

Table 13. Document Embeddings on Cora Link Prediction (micro-F_1)

Table 14 show results for classic methods for Link Prediction problem on CiteSeer dataset, surprisingly, here BOW and TF-IDF perform very poorly, whereas SBERT shows superior performance. SBERT performance makes a lot of sense since it is trained to be able to differentiate two texts between each other so it is incredibly good for such kind of tasks.

% Train edges	5%	10%	30%	50%
Bag of Words	0.52±0.01	0.52±0.00	0.52±0.01	0.52±0.00
LDA	0.69±0.01	0.69±0.01	0.70±0.01	071±0.01
SBERT pretrained	**0.84±0.00**	**0.85±0.00**	**0.86±0.01**	**0.86±0.01**
Word2Vec (d=300)	0.54±0.00	0.54±0.00	0.54±0.00	0.54±0.01
Doc2Vec (d=300)	0.77±0.01	0.77±0.00	0.78±0.00	0.79±0.00
Sent2Vec (d=600)	0.54±0.00	0.55±0.01	0.55±0.00	0.56±0.01

Table 14. Classic Document Embeddings on Citeseer-M10 Link Prediction (micro-F_1)

Generally, one can say that for Link Prediction choice of the Document Embeddings algorithm should be made per dataset as there is no universal best performer.

When one mask edges of a network, it changes the structure of the graph (contrary to Node Classification), so it might be more challenging for purely network embeddings method to perform well. Table 15 shows how network embedding algorithms perform on Cora for Link Prediction task.

% Train edges	5%	10%	30%	50%
DeepWalk	0.56±0.01	0.60±0.00	**0.66±0.00**	0.66±0.01
Node2Vec	**0.57 ±0.01**	**0.61±0.01**	0.65±0.01	**0.68±0.01**
HOPE	0.50±0.00	0.50±0.00	0.51±0.00	0.52±0.00

Table 15. Network Embeddings on Cora Link Prediction (micro-F_1)

For Citeseer-M10 dataset (Table 16) the situation is quite similar to Cora dataset in a sense that Node2Vec performs better than DeepWalk and both of these methods significantly outperforms HOPE. Results for DBLP are omitted but they are pretty much the same.

To sum these experiments up, for a Link Prediction problem (contrary to Node Classification) text information plays a much more important role than network structure.

% Train Edges	5%	10%	30%	50%
DeepWalk	**0.55±0.01**	**0.59±0.01**	0.66±0.01	0.66±0.00
Node2Vec	**0.55±0.00**	**0.59±0.00**	**0.69±0.00**	**0.71±0.00**
HOPE	0.50±0.00	0.51±0.00	0.54±0.00	0.57±0.00

Table 16. Network Embeddings on Citeseer-M10 Link Prediction (micro-F_1)

For the Link Prediction task, one can see (Table 17) that classic methods (BOW and TF-IDF) actually outperform more advanced combinations (Sent2Vec and Word2Vec). Also, TADW performs on the same level that plain document embeddings techniques. It might happen because TADW relies mostly on network information rather than on text.

% Train Edges	5%	10%	30%	50%
TADW - BOW	0.72±0.02	0.72±0.01	0.73±0.01	0.73±0.01
TADW - TF-IDF	0.73±0.02	0.74±0.01	0.74±0.01	0.75±0.01
TADW - Sent2Vec	0.70±0.01	0.70±0.01	0.71±0.00	0.73±0.00
TADW - Word2Vec	0.64±0.01	0.68±0.00	0.71±0.01	0.72±0.01
GCN - TF-IDF	**0.78±0.01**	**0.78±0.01**	**0.79±0.01**	**0.80±0.01**
GCN - Sent2Vec	0.69±0.01	0.71±0.01	0.73±0.01	0.75±0.01
GCN - SBERT	0.67±0.01	0.69±0.01	0.71±0.01	0.73±0.01
GCN (Custom)	0.72±0.01	0.75±0.01	0.75±0.01	0.75±0.01

Table 17. Custom modifications on Cora Link Prediction (micro-F_1)

On Citeseer-M10 dataset (Table 18) combination of GCN and SBERT shows by far the best results (especially when the percentage of training edges is high). It is quite predictable considering SBERT performance on this dataset. What is interesting though is the fact that using SBERT alone performs better than in combination with GCN.

% Train Edges	5%	10%	30%	50%
TADW - BOW	0.50±0.01	0.51±0.02	0.51±0.01	0.52±0.01
TADW - TF-IDF	0.51±0.01	0.51±0.01	0.51±0.01	0.52±0.01
TADW - Sent2Vec	0.52±0.01	0.53±0.00	0.53±0.00	0.54±0.00
TADW - Word2Vec	0.52±0.01	0.52±0.01	0.53±0.00	0.53±0.00
GCN - TF-IDF	**0.68±0.01**	0.69±0.01	0.70±0.01	0.70±0.01
GCN - Sent2Vec	0.59±0.01	0.62±0.01	0.67±0.01	0.68±0.01
GCN - SBERT	**0.68±0.01**	**0.70±0.01**	**0.72±0.01**	**0.77±0.01**
GCN (Custom)	0.61±0.01	0.67±0.00	0.68±0.01	0.68±0.01

Table 18. Custom modifications on Citeseer-M10 Link Prediction (micro-F_1)

As far as other fusion methods are concerned (Table 19) one can note that all methods except for GCN do not significantly outperform document embeddings techniques, which seem to provide a very solid baseline for Link Prediction. Also, GCN shows very consistent results for different percentage of train edges, whereas other methods seem to degrade heavily when the percentage of train nodes is not high.

On Citeseer (Table 20) most of the fusion methods perform quite poorly. The only method that shows a good performance is GCN with SBERT as document embeddings technique.

Generally, one can see that for Link Prediction task (as opposite to Node Classification problem) SBERT shows superior performance to other document embeddings technique when used alone and in combination with other methods (such as GCN).

4.3 Graph Visualization

The main goal of Graph Visualization is to produce a meaningful 2-D plot of nodes. Meaningful visualization would place nodes from one class close to each other and nodes from different classes far away from each other. Consequently, having solved the visualization problem, one automatically gets node clustering and vice versa.

In order to produce a 2-D plot, one has to find a vector of two points describing the position of a node. This problem can be solved in two ways using network embeddings:

1. Explicitly learn embeddings of size two using any methods described in the previous chapters.
2. First learn embeddings of length d, then use a dimensionality reduction method to get vectors of the size 2.

Here the second approach would be preferred since the first one generally produce worse results. Because it is a much harder task to learn a sensible representation of size 2. Therefore the embeddings dimension size would be borrowed from the previous experiments. The whole network will be fed into embeddings algorithms without any masking. Then obtained embeddings are to be mapped into 2-D space using T-SNE algorithm.

For all models, hyperparameters are chosen in the same way as in previous experiments. In the case of GCN activations of the first convolutional layer are used as graph embeddings.

% Train Edges	5%	10%	30%	50%
BOW + DeepWalk	0.70±0.01	0.73±0.01	0.76±0.00	0.76±0.01
Sent2Vec + DeepWalk	0.70±0.01	0.73±0.01	0.73±0.01	0.75±0.01
TADW - TF-IDF	0.73±0.02	0.74±0.01	0.74±0.01	0.75±0.01
TriDNR	0.71±0.01	0.73±0.00	0.74±0.01	0.76±0.01
GCN - TF-IDF	**0.78±0.01**	**0.78±0.01**	**0.79±0.01**	**0.80±0.01**
GCN (Custom)	0.72±0.01	0.75±0.01	0.75±0.01	0.75±0.01

Table 19. Fusion methods on Cora Link Prediction (micro-F_1)

% Train Edges	5%	10%	30%	50%
BOW + DeepWalk	0.56±0.01	0.60±0.00	0.66±0.00	0.66±0.01
Sent2Vec + DeepWalk	0.55±0.01	0.59±0.01	0.66±0.01	0.67±0.01
TADW - TF-IDF	0.73±0.02	0.74±0.01	0.74±0.01	0.75±0.01
TADW - TF-IDF	0.51±0.01	0.51±0.01	0.51±0.01	0.52±0.01
GCN - TF-IDF	**0.68±0.01**	0.69±0.01	0.70±0.01	0.70±0.01
GCN - SBERT	**0.68±0.01**	**0.70±0.01**	**0.72±0.01**	**0.77±0.01**

Table 20. Fusion methods on Citeseer Link Prediction (micro-F_1)

Analyzing results obtained without fusion (Figure 1) one can note that even simple TF-IDF method provides a very strong baseline - most of the classes (represented by colours) are clearly separable. There are some exceptions though, purple and pink classes are not concentrated. Sent2Vec shows even better results, samples of every class are tightly clustered together, the only problem is that clusters itself are very close to each other, which means that it would be rather hard to separate them using clustering algorithms. DeepWalk's visualization improves this aspect - different classes are placed quite far away from each other. However, some classes are separated themselves, for example, green and red classes clearly consists of two classes.

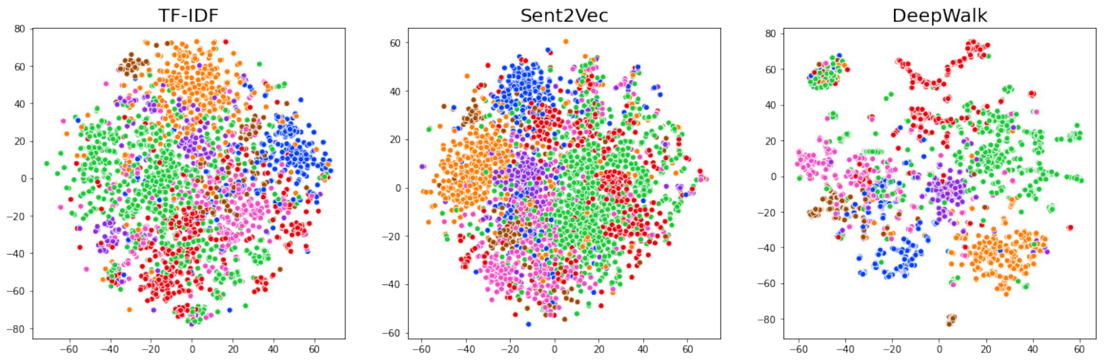

Figure 1. TF-IDF, Sent2Vec and DeepWalk embeddings visualization on Cora

Fusion methods (Figure 2) provide even better visualizations: TADW shows results that are very similar to DeepWalk but with better consistency across different classes. TriDnr is more similar to Sent2Vec - it provides a very clean separation of classes, but the classes themselves are located very close to each other, so it would be hard to apply clustering algorithms. GCN provides by far the best results - classes are far away from each other and clearly separated, so it would be very easy to make a clusterization.

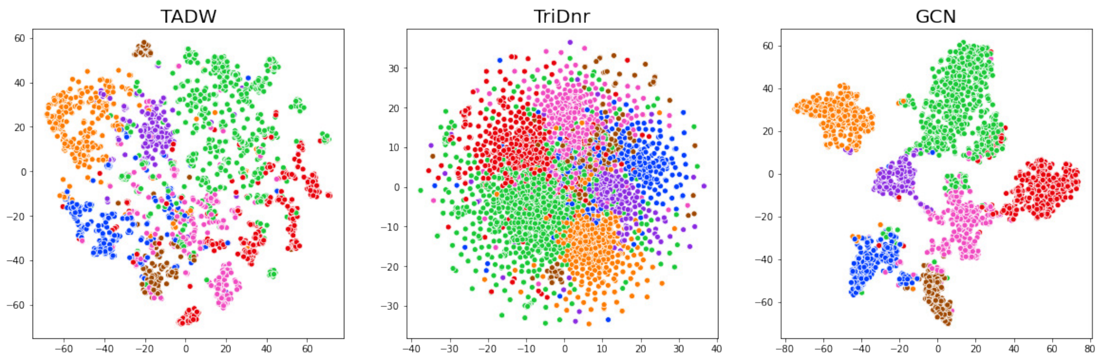

Figure 2. TADW, TriDnr and GCN embeddings visualization on Cora

5 DISCUSSION

One can see that fusion of graph and text information shows superior results for all Machine Learning tasks on graphs in comparison with methods that use only text or network information. It proves that text information and network structure are complementary to each other, but the contribution of each component depends on a task and a dataset. It is also clear that using advanced document embedding techniques such as Sent2Vec and SBERT can significantly boost the performance of fusion methods such as TADW and GCN. The reason for that is the fact that advanced document embeddings can better capture semantic of the words (synonyms, antonyms, etc.) and therefore better generalize. What is more pre-trained embeddings might be preferable when the number of training samples is low. However, Bag of Words and TF-IDF provide a very solid baseline for Machine Learning tasks on citation networks because nodes (scientific papers) can be efficiently represented by a set of keywords. The choice of the document embedding technique should be task-dependent: SBERT works better for Link Prediction, whereas Sent2Vec shows good performance for Node Classification. It can be explained by the fact that Sent2Vec is aimed to restore overall semantic of a document, whereas SBERT is specifically trained to predict whether two documents describe the same thing or not, so it is no wonder that SBERT is incredibly good for the Link Prediction. Although custom GCN architecture, which allows to simultaneously learn word and network embeddings, do not outperform state of the art algorithms, still there is some potential: it might be a sensible idea to pretrain text embeddings layer first and then finetune convolutional layers, also it might show much better results on bigger networks (millions of nodes) as it takes a lot of data to train good word embeddings.

6 CONCLUSION

In this work, a comprehensive comparison of different fusion methods for Machine Learning problems on graphs was conducted. The best combinations of network and document embeddings for different Machine Learning tasks are outlined and compared with the traditional approaches. The new GCN architecture is proposed to be able to simultaneously learn document and network representations.

Main conclusions of the work:

1. Fusion of text and graph information allows to significantly boost performance on Machine Learning tasks.

2. Usage of advanced document embeddings such as Sent2Vec and SBERT can improve the accuracy of different fusion architectures such as TADW and GCN. SBERT generally works better for Link Prediction, Sent2Vec for Node Classification.

3. There is no universal solution that fits all problems and all datasets. Different methods (and combinations of methods) might work better for different datasets.

4. Proposed GCN modification does no work well for datasets considered in this work but might show a better performance for bigger networks with more text.

This work might be extended in the future in the following ways: firstly, it is promising to experiment with the proposed GCN architecture using bigger networks (ideally millions of nodes), it might show better results because a lot of data is required to learn sensible word embeddings. Another possible extension is to use a joint loss to simultaneously learn network and text embeddings. For instance, combining GCN and BERT might present very competitive results for Link Prediction.

REFERENCES

Barabási, A.-L. and Pósfai, M. (2016). *Network science*. Cambridge University Press, Cambridge.

Belkin, M. and Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Brownlee, J. (2020). One-vs-rest for multi-class classification.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653.

Harris, Z. S. (1954). Distributional structure. *Word*, 2–3(10):146–162.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks.

Lau, J. and Baldwin, T. (2016). An empirical evaluation of doc2vec with practical insights into document embedding generation. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 78–86.

Lim, K. W. and Buntine, W. L. (2016). Bibliographic analysis with the citation network topic model. *CoRR*, abs/1609.06826.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T. and Le, Q. (2014). Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188—-1196.

Ou, M., Cui, P., Pei, J., Zhang, Z., and Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1105–1114, New York, NY, USA. Association for Computing Machinery.

Pagliardini, M., Gupta, P., and Jaggi, M. (2017). Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*.

Pagliardini, M., Gupta, P., and Jaggi, M. (2018). Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. In *NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics*.

Pan, S., Wu, J., Zhu, X., Zhang, C., and Wang, Y. (2016). Tri-party deep network representation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 1895–1901. AAAI Press.

Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, 5(24):513–523.

Sen, P., Namata, G. M., Bilgic, M., Getoor, L., Gallagher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, 29(3):93–106.

Yang, C., Liu, Z., Zhao, D., Sun, M., and Chang, E. Y. (2015). Network representation learning with rich text information. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 2111–2117. AAAI Press.

Yang, J. and Leskovec, J. (2015). Defining and evaluating network communities based on ground-truth. *Knowl. Inf. Syst.*, 42(1):181–213.