On Joint Representation Learning of Network Structure and Document Content

Jörg Schlötterer(✉), Christin Seifert, and Michael Granitzer

University of Passau, Innstraße 33, 94032 Passau, Germany
{joerg.schloetterer,christin.seifert,michael.granitzer}@uni-passau.de
http://www.uni-passau.de

Abstract. Inspired by the advancements of representation learning for natural language processing, learning continuous feature representations of nodes in networks has recently gained attention. Similar to word embeddings, node embeddings have been shown to capture certain semantics of the network structure. Combining both research directions into a joint representation learning of network structure and document content seems a promising direction to increase the quality of the learned representations. However, research is typically focused on either word or network embeddings and few approaches that learn a joint representation have been proposed. We present an overview of that field, starting at word representations, moving over document and network node representations to joint representations. We make the connections between the different models explicit and introduce a novel model for learning a joint representation. We present different methods for the novel model and compare the presented approaches in an evaluation. This paper explains how the different models recently proposed in the literature relate to each other and compares their performance.

Keywords: Representation learning · Network embeddings · Document embeddings

1 Introduction

Recently, there has been an uptake of the representation learning methods from the domain of natural language processing to network structures. Network representation learning addresses the sparsity issue commonly faced by machine learning applications in networks by representing nodes in a unified low-dimensional space. These node representations have been shown to retain certain properties of the network structure, such as homophily or structural equivalence [3]. However, in most networks additional information beyond the network structure is associated with a vertex, such as textual content or meta data. Consider for example a paper citation network as depicted in Fig. 1. In this network, the additional information associated with a node is textual information, i.e., the document content. Even further, each node is a vertex in the graph and a document at once.

A. Holzinger et al. (Eds.): CD-MAKE 2017, LNCS 10410, pp. 237–251, 2017.
DOI: 10.1007/978-3-319-66808-6_16

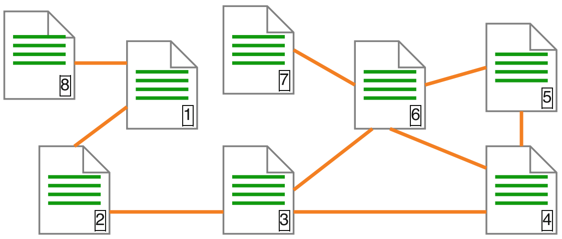


Fig. 1. Paper citation network with **textual information** (document content) and **link information** (citations) (Color figure online)

Both types, the network node and the document, can be represented as real-valued vectors with fixed length, so called embeddings. These embeddings can then be utilized as input features for subsequent tasks like node classification or link prediction see e.g. [3,11]. The goal of node (or document) classification in the citation network is to assign a particular category to a document (node), in order to organize a document collection. Link prediction in the context of a paper citation network refers to identifying relevant papers, which have not been cited in the paper at hand. Reasons for relevant papers not being cited are for example space constraints or authors publishing at the same time and hence not knowing about each others work.

For the citation network in Fig. 1, embeddings can be obtained by different means: Based on the textual document content, embeddings can be obtained via the Paragraph Vector (PV) [7] approach. An alternative way is to use the network or link information, for example with Spectral Clustering [12]. In this paper, we focus on node embeddings, like Deepwalk [9], LINE [11] or Node2Vec [3], that build on the representation learning techniques from the natural language processing domain. Only few approaches have been proposed that combine both modalities, link and text information. To the best of our knowledge, TADW [13] and Paper2Vec [2] are the only methods that combine both modalities using representation learning.

In this paper, we first provide background information about word and document embeddings and their adaption to graph structures. From this, we derive a novel baseline approach for a joint embedding of textual content and network structure. The assumption for the joint embeddings is that they retain information from both modalities (textual content and network structure). We assume these joint embeddings to be richer in expressiveness than representations trained on either text or network alone and hence provide a higher quality representation for subsequent tasks like node classification or link prediction. Then we review related work and present an evaluation of the different approaches (text only, graph only and combined). We conclude with an outlook on future work.

2 Background

In this section, we present models that use shallow neural networks to create the vector representations. We start with word embeddings, followed by their extension to document embeddings and their adaptation to graph structures. The common principle for all these models is the distributional hypothesis "you shall know a word by the company it keeps" [1], which states that words in similar contexts tend to have similar meanings [4]. That is, similar words (or nodes) tend to appear in similar word (node) neighborhoods.

2.1 Word Embeddings

The approach to compute continuous vector representations of words proposed by Mikolov et al. [7] became famous as Word2Vec. A sliding window is moved across a sentence, in order to define the context for a word. This process is illustrated in Fig. 2. The words, considered as context are defined by the window size, which is the maximum distance between the word under consideration and a context word. In our example, the window size is 2, i.e., 2 words to the left of $w(t)$, the word under consideration, and 2 words to the right are considered as context: The context words for *fox* are *quick*, *brown*, *jumped*, *over*.

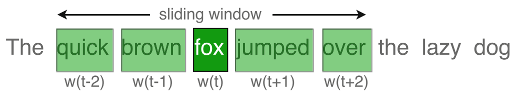


Fig. 2. Illustration of word embedding sliding window with window size 2

Mikolov et al. proposed two model architectures to compute the embeddings: The continuous bag-of-words (CBOW) model, illustrated in Fig. 3a and the continuous skip-gram (SG) model, illustrated in Fig. 3b. While CBOW tries to predict a word, given the context, SG tries to predict the context, given a word. Both models use a three-layer neural network, with linear activation in the hidden layer and a softmax in the output layer. The input in the SG model is a one-hot-coding and similarly in the CBOW model, a k-hot-coding activating the k inputs of the words in context. The output of the hidden layer in the CBOW model are the averaged weights between the activated inputs and the hidden layer. That is, only the weights of the k activated inputs are considered. Similarly, in the SG model, only the weights from the input word are considered. No averaging is needed in this case and hence, the output of the hidden layer is just a copy of the weights between the active input and the hidden layer. Furthermore, the SG model is trained by pairs of word and context word. The training pairs within the context window of our example from Fig. 2 are: (fox, quick), (fox, brown), (fox, jumped) and (fox, over). As an optimization step, Mikolov et al. suggest to give less weight to more distant context words, since

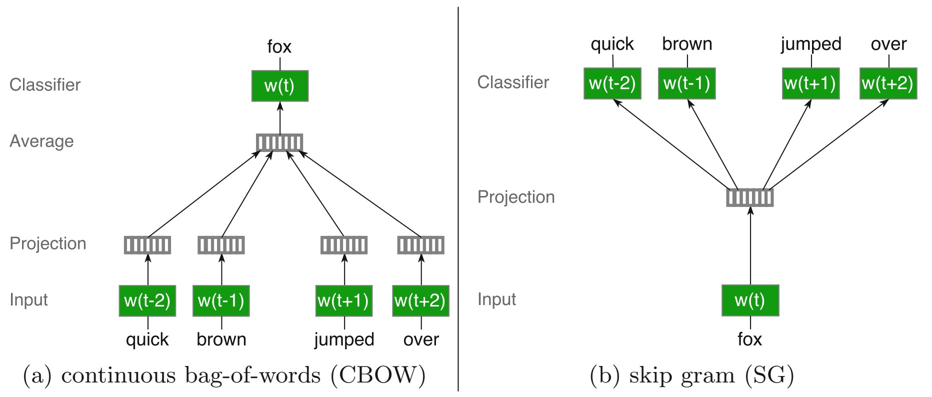


Fig. 3. Illustration of the two model architectures of Word2Vec (CBOW and SG) with the sliding window context from the example sentence in Fig. 2. The projection corresponds to the weights between input and hidden layer, i.e., the word embeddings.

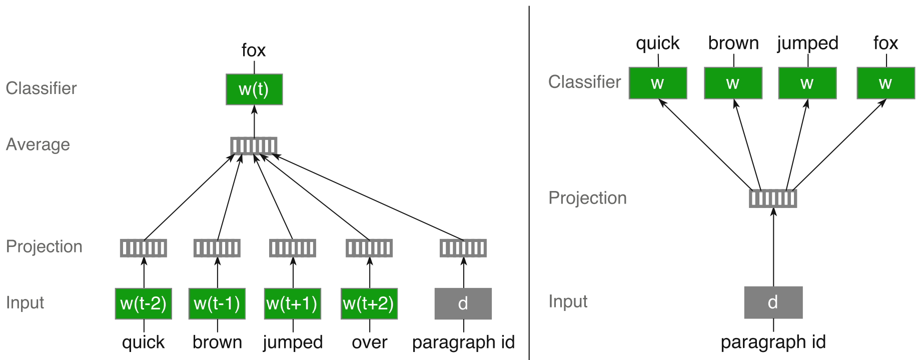
more distant words are usually less related to the current word. This is achieved by not always sampling pairs from the complete window, but sampling the pairs from a randomly smaller window. The final embedding vector of a word is given by the weights between its corresponding input neuron and the hidden layer.

2.2 Document Embeddings

With Paragraph Vector [5], Quoc and Mikolov extended the Word2Vec approach to sentences and documents (also known as Doc2Vec). Therefore, they extended the two architectures presented in the previous section with an additional paragraph identifier, yielding the distributed memory (PV-DM) model, illustrated in Fig. 4a and the distributed bag-of-words (PV-DBOW) model, illustrated in Fig. 4b.

While PV-DM extends the CBOW model, PV-DBOW corresponds to the SG model. In PV-DM, a sliding window is moved across the paragraph (or document), similar to sliding a window across the sentences in Word2Vec. In contrast to Word2Vec, an additional paragraph identifier is added to the context. This paragraph identifier can be thought of as another word, which acts as a memory that remembers what is missing from the current context – or the topic of the paragraph. Therefore, it got named distributed memory. PV-DBOW uses the paragraph identifier to predict the context words. Here, context words are the words contained in the paragraph. The model is trained by pairs of (<paragraph_id>, <word in paragraph>). In the figure, we illustrated only four context words, i.e., four training pairs. Of course, during training, all of the words in a paragraph are to be considered.

It is to note that the paragraph identifier is just a symbolic token, which carries no semantic meaning in itself. However, the embedding of this identifier, i.e. the weights between the corresponding input neuron and the hidden layer



(a) distributed memory (PV-DM)   (b) distributed bag-of-words (PV-DBOW)

Fig. 4. Illustration of the two model architectures (PV-DM and PV-DBOW) of paragraph vector (Doc2Vec). The left part (PV-DM) shows the sliding window context from Fig. 2. PV-DBOW (right part) does not use a sliding window, instead all words from the sentence are considered as context.

(indicated as "projection" in the figure), does: After training the neural network, documents with similar content should have a similar representation in the embedding space, i.e., their weight vectors (projections) should be similar.

2.3 Graph Embeddings

With Deepwalk [9], Perozzi et al. adapt the Word2Vec approach for learning vector representations of vertices in a network. Therefore, they sample random walks for each node in the network and apply the skip gram (SG) model on the sampled walks. Those random walks can be thought of as the equivalent of sentences in the Word2Vec approach, with nodes resembling words. Accordingly, a sliding window is moved across the sampled walks to define the context. The analogies between nodes as words and walks as sentences apply only on the conceptual level. The actual input to the Deepwalk algorithm is in fact a graph. In our case, it is a citation network, as illustrated in Fig. 1, in which nodes correspond to papers and edges are given by the citations.

The sliding window across a walk is illustrated in Fig. 5. The figure is the graph equivalent to Fig. 2, the sliding window across a sentence. The random walk was sampled from the graph in Fig. 1. Perozzie et al. applied the SG model to train the vector representations, i.e., the current node predicts the nodes in the context. Alternatively, the CBOW model can also be applied to train the vectors, i.e., the nodes in the context predict the current node. We refer to the skip gram (SG) model on graphs as DW-SG and to the continuous bag-of-words (CBOW) model on graphs as DW-CBOW.

With Node2Vec [3] an extension to Deepwalk has been proposed, that allows to guide the walks in their exploration of the network. That is, the sampling

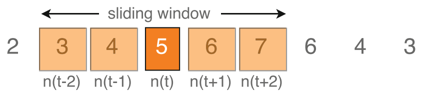


Fig. 5. Illustration of graph embedding sliding window with window size 2

procedure can be parametrized, such that the walks are biased towards the local neighborhood or tend to move further away (resembling breadth-first or depth-first search in the most extreme cases). The parameters, introduced to guide the walk are p and q. The parameter p controls the tendency to re-visit already visited nodes and q controls, whether the walk tends to move further away from the original node. In the evaluation, we used Node2Vec with a parameter setting that resembles the random walk sampling of Deepwalk ($p = 1, q = 1$). We refer to it as N2V. Typically, the SG model is used to train vector representations of nodes and we are also using that model, that is, we use N2V and N2V-SG synonymously.

3 Combining Link and Text Information

In this section, we present a novel model to learn combined embeddings for documents/nodes, fusing link and text information. We first show, how the Paragraph Vector approach (PV-DM and PV-DBOW as presented in Sect. 2.2 *document embeddings*) can be adapted to graph structures and then, how it can be extended to include textual information.

3.1 Paragraph Vector on Graphs

In Deepwalk, the Word2Vec approach is adapted to learn node representations. As a pre-requisite for our model, we show how the Paragraph Vector approach can also be adapted to represent nodes. Therefore, we assign the start node of each random walk as the paragraph identifier and treat the rest of the walk as paragraph content. That means, we sample a sub-graph around the starting node and learn an embedding vector for this sub-graph. If the walk length is chosen equal or larger than the network diameter, the random walks may cover the whole network. That is, the sampled sub-graph may not be a real sub-graph, but contain all nodes from the original graph.

On graphs, PV-DBOW is similar to DW-SG, if the walk length of PV-DBOW equals the window size of DW-SG. In this case, both models have the same context radius. In PV-DBOW, the walk length defines the maximum distance between the start node and nodes on the sampled walks. The random walks cannot contain nodes, that are further away from the start node than the walk length. Similarly, in DW-SG, the window size defines the maximum distance between the node under consideration and nodes to be considered as context. As those are sampled with random walks without jump probabilities, nodes that require more steps than the window size to be reached, are outside the context. It is to note, that while Deepwalk also has the walk length parameter, this parameter can be chosen to be larger, as the context radius is constrained afterwards by the window size.

3.2 Fusing Link and Text Information

We now have means to apply the Paragraph Vector model on both modalities, text and link information. This leads us to a combined model, by simply sharing the paragraph identifier between the text and the graph model. This is possible, as the paragraph identifier is just a symbolic identifier of the document or sub-graph respectively. We illustrate the combined model in Fig. 6. The illustrated model corresponds to the PV-DM approach, hence we refer to this architecture as text graph distributed memory (TG-DM). The left part of the figure is completely equivalent to PV-DM applied to textual content (cf. Fig. 4a), while the right part is equivalent to PV-DM for a sampled sub-graph (i.e. the set of walks with the same start node). These two models are combined via the shared weights of the paragraph identifier. This paragraph identifier is just a symbolic identifier, hence the "2" can be thought of as a document id and at the same time as a node id.

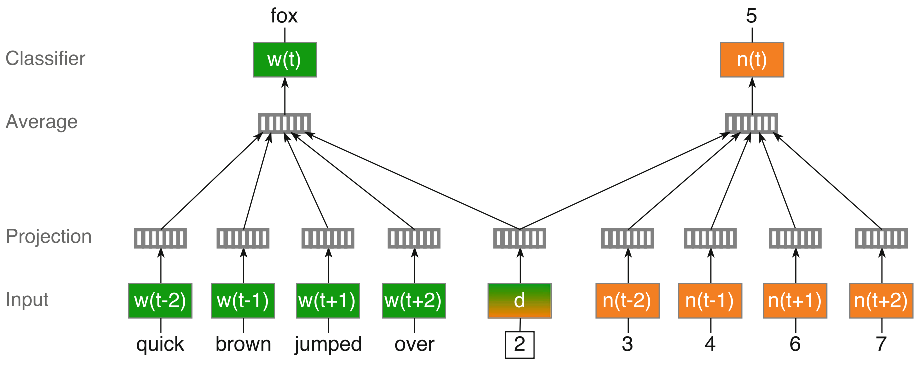


Fig. 6. Illustration of text graph distributed memory (TG-DM) model. The paragraph vector d is shared between the text (left part) and the network model (right part), acting as distributed memory for both of them at the same time.

The combination of text and link information with the PV-DBOW approach is achieved in the exact same fashion, by sharing the paragraph identifier. Hence we refer to the combination via PV-DBOW as text graph distributed bag-of-words (TG-DBOW). The term *words* in this case also includes network nodes, because walks are considered as sentences in a document. We investigated different methods to train the TG-DBOW model, namely: TG-MIX, TG-SPLIT and TG-SUM, which we present in the following.

TG-DBOW uses the skip gram architecture and hence, the training consists of a paragraph identifier and context words and nodes. The simplest method is then to use the architecture of PV-DBOW and treat the context nodes from the sampled random walks as words in the document. Again, the paragraph identifier is just a symbolic token, hence we can use the same identifier for the node and document. We call this method TG-MIX, which is depicted in Fig. 7.

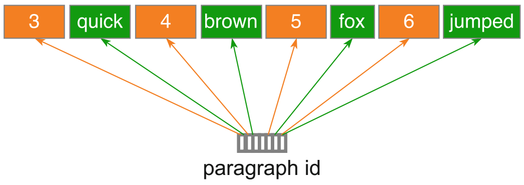


Fig. 7. TG-MIX

Another way to learn a combined embedding via TG-DBOW is to separate the output layers for nodes and words. We call this approach TG-SPLIT, depicted in Fig. 8. In this case, we alternate between showing the network paragraph identifier and context word pairs, and paragraph identifier and context node pairs. While the weights for the paragraph identifier are shared, the prediction of a context word or node is separated by using two separate softmax layers. These separate softmax layers are indicated with the two separate error measures E_1 and E_2 in the figure.

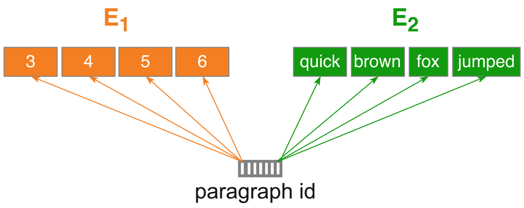


Fig. 8. TG-SPLIT

The third method, which we call TG-SUM, is depicted in Fig. 9. Here, we train the network by providing triples of paragraph identifier, context word and context node. That is, for a given paragraph identifier, the network simultaneously has to predict the context word and the context node. In contrast to TG-SPLIT, where in each alternating step the error to be propagated back is either E_1 or E_2, the error in TG-SUM is the sum of both errors $(E_1 + E_2)$ in every step. In preliminary experiments, TG-SPLIT outperformed TG-MIX and TG-SUM, hence we included only TG-SPLIT in the evaluation.

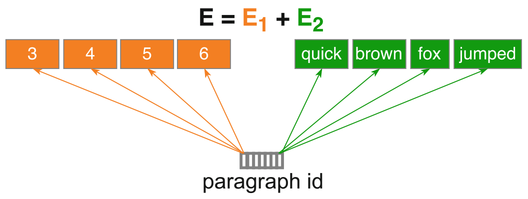


Fig. 9. TG-SUM

4 Related Work

In this section, we present two closely related approaches: TADW - Text Associated Deepwalk [13] and Paper2Vec [2]. To the best of our knowledge, these are the only two approaches that build upon the representation learning techniques from the natural language processing domain and combine text and link information into a single embedding.

4.1 Paper2Vec

With Paper2Vec, Ganguly and Pudi [2] do not propose a completely new model, but rather combine and extend existing models in a clever way. Paper2Vec is modeled around a citation network, i.e., the data for this approach needs to have at least a similar structure (documents with textual content, that are connected via links). Their contribution to improve the learned representations is twofold: First, they enrich the citation graph with artificial edges, based on the similarity of documents and second, they initialize the node embeddings with the paragraph vector representations obtained from the documents.

The intuition behind the enrichment of the citation graph with artificial edges is that those are links, which should actually be there, but are omitted because of the following potential reasons: Authors may not be fully aware of all the current work in the field and hence, might miss to cite some papers. Also, space constraints limit the amount of citable papers and trivially, two papers might appear at the same time and therefore the authors may not know about each others work. Adding artificial edges between similar documents can then be considered as adding the missing links. To this end, the authors first compute an embedding for each paper with the PV-DM approach. Then, they add artificial links between each document and its k nearest neighbors, based on their similarity in the document embedding space.

To create the final embeddings, Paper2Vec applies the Deepwalk algorithm on the enriched graph that contains both, the original and artificial edges. Instead of randomly initializing the vector representations of the nodes in this graph, the authors initialize them with the document embeddings learned in the previous step. This raises the interesting question, whether this initialization avoids start configurations, that would result in a local optimum or whether the initialization preserves information from the document content, that cannot be learned via the network structure.

4.2 TADW

As a first important step of the TADW approach, Yang et al. [13] showed that Deepwalk factorizes a matrix of transition probabilities, similar to Word2Vec factorizing a matrix of co-occurrences. More precisely, each entry in the matrix that gets factorized with Deepwalk is the logarithm of the average probability that vertex v_i randomly walks to vertex v_j in a fixed number of steps. Similarly, depending on the network architecture, an entry in the matrix that gets factorized by Word2Vec is the logarithm of the weighted co-occurrence between a (word, context word)-pair [8] or the Shifted Positive Pointwise Mutual Information (SPMI) of the pair [6]. The matrix factorization of Deepwalk is illustrated in Fig. 10. The vertex representation of Deepwalk, i.e., the node embeddings, is given by the matrix W. The second step of TADW is then to incorporate text information. Instead of factorizing M into a product of two matrices, TADW factorizes M into a product of three matrices, where the third factor is a matrix of text features. This process is illustrated in Fig. 11. The text feature matrix T is a low-dimensional representation of the tf-idf matrix of the document contents, obtained via singular value decomposition. The matrices W and H are obtained by alternately optimizing them. The final vector representation is obtained by concatenating W and $H \times T$, resulting in a 2d-dimensional embedding.

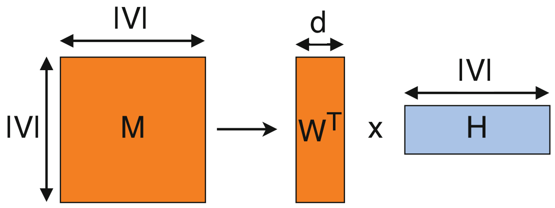


Fig. 10. Matrix factorization of Deepwalk.

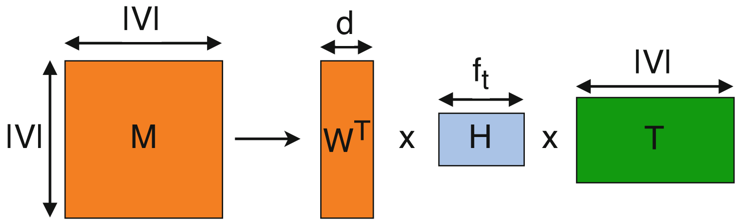


Fig. 11. Matrix factorization of Text Associated Deepwalk (TADW).

5 Evaluation

We evaluated[1] the aforementioned models on a node classification task on the CORA-ML dataset[2]. This dataset contains 2708 scientific publications from the machine learning domain. Each paper is assigned to one of seven classes (e.g. neural networks or reinforcement learning). The citation network of these publications consists of 5429 links. Each document in the dataset is described as a binary vector of 1433 dimensions, indicating the presence of the corresponding word. The document contents are short texts, generated from title, containing 18 words on average. On this dataset, node classification is equivalent to document classification. A label is assigned to a node or document respectively, while both represent the same thing (a document is at the same time a node in the citation network).

For a quick reference, we list the evaluated approaches again with a short description. The text-only models are colored green, the network-only models orange and the joint models blue.

PV-DBOW. The distributed bag-of-words model of paragraph vector [5]. Document embeddings are trained by predicting the context, i.e., the words in the corresponding document.

PV-DM. The distributed memory model of paragraph vector [5]. Predicts a word from the context words together with a paragraph identifier. The paragraph identifier acts as a memory for the context and is the document embedding.

DW-SG. Deepwalk with the skip gram model [9]. Node embeddings are trained by predicting the context, i.e., the neighboring nodes on a sampled random walk.

N2V. According to the authors, with parameters ($p = 1$, $q = 1$), N2V resembles DeepWalk [3]. We used it as an alternative to DW-SG in the evaluation.

CC. Naive baseline concatenation of document (PV-DBOW) and node embeddings (N2V).

TG-DM The paragraph vector model with distributed memory combining link and text information.

TG-SPLIT. The paragraph vector model with the skip gram split approach, i.e. two separate output layers in the corresponding neural network.

P2V. The Paper2Vec approach, adding artificial links and initializing node with document embeddings [2].

TADW. Text associated deepwalk, factorizing the matrix of transition probabilities into three matrices [13].

We learned the vector representations on the whole dataset. Then we trained a support vector machine (SVM) on different portions of the data to predict the class in a supervised fashion, with the vector representations as input features.

[1] evaluation scripts are available at https://github.com/schloett/tg-split.

[2] http://www.cs.umd.edu/~sen/lbc-proj/LBC.html.

We opted for an SVM as it was also used in the related work for the node classification task [2,13]. The training portions were varied in 10% steps from 10% to 90%. We averaged the accuracy over 20 random splits for each percentage. An overview with the average accuracy across all training portions is given in Table 1. We were not able to run the Matlab implementation of TADW, due to issues with the mex-file, therefore, we had to exclude TADW from this comparison. The detailed results with averaged accuracy over the random splits for every percentage is given in Table 2. We explored different hyper-parameter settings with Bayesian Optimization [10], but did not find a major improvement. Therefore, we decided to stick to similar parameters as reported in the related work. We set the dimension of all the trained vector representations to 100. For the models including network information, we sampled 10 walks with a length of 40 for each node. We chose a window size of 5 and for models with negative sampling, we set the amount of samples to 10. As TG-SPLIT performed better than TG-SUM and TG-MIX in preliminary experiments, we only included TG-SPLIT in the evaluation.

Table 1. Results overview, with averaged accuracy over all training portions.

Text-only		Network-only	Joint			
PV-DM	**PV-DBOW**	**N2V**	**CC**	**TG-DM**	**TG-SPLIT**	**P2V**
53.5	65.4	83.4	84.8	54.7	83.1	78.7

Beyond the results from our evaluation, the detailed Table 2 contains the results reported in the original papers of TADW and Paper2Vec for comparison. The column "source" indicates the paper, from which the results were copied and the corresponding rows are colored with a light gray background. Both papers reported accuracy scores only up to a training percentage of 50%. It is to note, that the document contents used for evaluation in Paper2Vec differ from the original CORA-ML dataset. In Paper2Vec, the authors used the full text of the documents, while the original dataset contains less information (18 words per document on average). This explains the rather large difference in the text-only baselines[3] and also has an influence on the other methods, that take the document content into account. Hence the results are not directly comparable. Further, the evaluation was carried out on random splits of the data (20 splits in our experiments, 10 splits in Paper2Vec and TADW). The actual splits are of course not the same, but the differences in the averaged results are less significant than the aforementioned differences in the data.

As can be seen from the table, the approach for combining link and text information presented in Sect. 3 does not outperform the naive baseline of concatenating node and document embeddings. Also, its performance is close to the

[3] Even though the tf-idf dimensions are the same (1433) they contain different information. For tf-idf in Paper2Vec, the authors "kept the maximum features (sorted by df value) as 1433".

Table 2. Detailed results, with averaged accuracy over random splits for every training percentage. Light gray rows indicate results reported in the paper corresponding to source. The use of full text documents in the evaluation data is indicated by *. Text-only approaches are colored green (upper part), network-only are colored orange (middle part) and combined approaches are colored blue (lower part) and further separated by two lines.

	model	SVM training ratio								
source	(dimensions)	10%	20%	30%	40%	50%	60%	70%	80%	90%
	PV-DM (100)	43.3	48.9	51.8	53.7	54.9	56.2	56.7	57.7	58.3
	PV-DBOW (100)	57.9	62.3	64.4	65.7	66.7	67.2	67.7	68.3	68.6
p2v*	PV-DM (100)	76.8	81.1	82.8	83.5	84.4	n.a.	n.a.	n.a.	n.a.
	tf-idf (1433)	37.9	47.8	55.6	61.6	65.2	68.4	70.7	72.0	73.4
p2v*	tf-idf (1433)	78.5	82.8	84.7	86.0	86.9	n.a.	n.a.	n.a.	n.a.
tadw	tf-idf (200)	58.3	67.4	71.1	73.3	74.0	n.a.	n.a.	n.a.	n.a.
	N2V (100)	77.2	80.7	82.5	83.6	84.5	84.8	85.3	85.8	85.9
tadw	DW-SG (100)	76.4	78.0	79.5	80.5	81.0	n.a.	n.a.	n.a.	n.a.
p2v*	DW-SG (100)	76.0	80.7	82.7	84.3	85.3	n.a.	n.a.	n.a.	n.a.
	P2V (100)	75.3	77.1	78.2	78.8	79.3	79.8	80.0	80.2	79.9
p2v*	P2V (100)	83.4	86.5	87.5	88.3	88.9	n.a.	n.a.	n.a.	n.a.
	TG-DM (100)	43.5	48.3	51.8	54.1	56.1	57.7	59.1	60.5	61.5
	TG-SPLIT (100)	73.4	79.7	82.3	83.9	84.9	85.5	85.9	86.1	86.3
	CC (200)	78.4	82.0	83.7	84.9	85.8	86.4	86.8	87.3	87.6
p2v*	CC (200)	80.6	83.8	85.1	86.4	87.1	n.a.	n.a.	n.a.	n.a.
tadw	CC (300)	76.5	80.4	82.3	83.3	84.1	n.a.	n.a.	n.a.	n.a.
p2v*	TADW (160)	82.4	85.0	85.6	86.0	86.7	n.a.	n.a.	n.a.	n.a.
tadw	TADW (200)	82.4	85.0	85.6	86.0	86.7	n.a.	n.a.	n.a.	n.a.

performance of node embeddings alone and inferior to TADW. With our model showing no improvement over the concatenated baseline, we assume that it is not able of making use of text information to complement link information and vice versa: Intuitively, we aim for similar embeddings of papers that have either similar content, similar citations or both. That is, papers with similar content should have a similar representation in the embedding space, regardless of their link structure (accordingly for papers with similar citations). This similarity criterion does not seem to be reflected in the embeddings, potentially because the optimization objective during learning the embeddings is to predict the context (or to predict from the context), not the class label.

We were not able to reproduce the high accuracy reported by Paper2Vec, which we account in particular to the differences in the dataset used for evaluation: Paper2Vec used the full text of the document, while the documents in the original CORA-ML dataset, which we used are short texts (18 words on average). Also, we may not have found the optimal set of hyper-parameters for the P2V approach. The accuracies of P2V are even lower than N2V. This may be caused by the lower quality document embeddings (cf. PV-DM vs. p2v* PV-DM), as those are used to add artificial links to the graph. It seems that more noisy links are added, lowering the performance of the node embeddings trained on the enriched network.

Summarizing the results, the novel baseline TG-SPLIT does not improve over the naive baseline of simply concatenating the node and document embeddings. In fact, its performance is comparable to the network-only model N2V. This indicates that the joint representation TG-SPLIT cannot make use of the mutual benefit, that we expected from incorporating network structure and document content.

6 Summary and Future Work

In this paper, we presented an overview on representation learning for combined network and text information, based on recent representation learning models from the natural language domain. Therefore, we gave a summary of the most prominent word embedding model and its extension to documents. We also summarized approaches, that adapt the word embeddings for network learning. We showed how document embeddings can be adapted to graph structures and introduced a novel baseline for a combined learning. We evaluated the presented models on a paper citation network. Results show, that the introduced baseline TG-SPLIT does not improve over the naive baseline of concatenating node and document embeddings and is comparable to the network-only model N2V. That is, TG-SPLIT does not seem to be able to connect network structure to document content for a mutual benefit.

In future work, we aim to investigate the question raised in Sect. 4.1 on whether the initialization of node embeddings with document embeddings avoids local optima or transfers information from the document content to the node representation (or whether it is a combination of both and how the influence of each is). We further aim to investigate other neural network architectures for the combined learning, in order to find a bridge between the text and the link information, similarly to TADW.

References

1. Firth, J.R.: A synopsis of linguistic theory 1930–55. In: Studies in Linguistic Analysis (Special Volume of the Philological Society), vol. 1952–1959, pp. 1–32. The Philological Society, Oxford (1957)
2. Ganguly, S., Pudi, V.: Paper2vec: combining graph and text information for scientific paper representation. In: Jose, J.M., Hauff, C., Altıngovde, I.S., Song, D., Albakour, D., Watt, S., Tait, J. (eds.) ECIR 2017. LNCS, vol. 10193, pp. 383–395. Springer, Cham (2017). doi:10.1007/978-3-319-56608-5_30

3. Grover, A., Leskovec, J.: Node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016, NY, USA, pp. 855–864. ACM, New York (2016)
4. Harris, Z.: Distributional structure. Word **10**(23), 146–162 (1954)
5. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of the 31st International Conference on Machine Learning, PMLR, Bejing, China, 22–24 June 2014, pp. 1188–1196 (2014)
6. Levy, O., Goldberg, Y.: Neural word embedding as implicit matrix factorization. In: Proceedings of the 27th International Conference on Neural Information Processing Systems, NIPS 2014, pp. 2177–2185. MIT Press, Cambridge (2014)
7. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Proceedings of the 26th International Conference on Neural Information Processing Systems, NIPS 2013, pp. 3111–3119. Curran Associates Inc., USA (2013)
8. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
9. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014, NY, USA, pp. 701–710. ACM, New York (2014)
10. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Proceedings of the 25th International Conference on Neural Information Processing Systems, NIPS 2012, pp. 2951–2959. Curran Associates Inc., USA (2012)
11. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, WWW 2015, pp. 1067–1077. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2015)
12. Tang, L., Liu, H.: Leveraging social media networks for classification. Data Min. Knowl. Discov. **23**(3), 447–478 (2011)
13. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.Y.: Network representation learning with rich text information. In: Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI 2015, pp. 2111–2117. AAAI Press (2015)