Paper2vec: Combining Graph and Text Information for Scientific Paper Representation

Soumyajit Ganguly and Vikram Pudi

International Institute of Information Technology Hyderabad, India.
soumyajit.ganguly@research.iiit.ac.in,
vikram.iiit.ac.in

Abstract. We present Paper2vec, a novel neural network embedding based approach for creating scientific paper representations which make use of both textual and graph-based information. An academic citation network can be viewed as a graph where individual nodes contain rich textual information. With the current trend of open-access to most scientific literature, we presume that this full text of a scientific article contain vital source of information which aids in various recommendation and prediction tasks concerning this domain. To this end, we propose an approach, Paper2vec, which comprises of information from both the modalities and results in a rich representation for scientific papers. Over the recent past representation learning techniques have been studied extensively using neural networks. However, they are modeled independently for text and graph data. Paper2vec leverages recent research in the broader field of unsupervised feature learning from both graphs and text documents. We demonstrate the efficacy of our representations on three real world academic datasets in two tasks - node classification and link prediction where Paper2vec is able to outperform state-of-the-art by a considerable margin.

Keywords: citation networks, representation learning, text and graph

1 Introduction and Related Work

Information mining from citation networks is a well studied problem but most research in this direction tackles it as a graph problem. This creates an outright loss of information as we drop the entire textual content from papers and consider them merely as nodes in a heterogeneous graph. Today, especially in the domain of Computer Science, almost all of the published full-length research articles are freely available from online websites like CiteSeerX and arXiv. This leaves us with ample opportunity to exploit the text information from these scientific papers. Being limited to merely the citation information in the graph can have drawbacks. While writing a paper, authors always have a space constraint and thus can only cite a limited number of prior literature. It is also not possible for a particular author to know or track each and every related research work to hers from this growing sea of knowledge. Often it happens that multiple leading-edge research being done on the same problem statement are unable to cite each other due to the close proximity of publication dates. All of these problems create more sparsity in the citation graph and we lose out on probable valuable edges.

In classical literature a text document is represented by its histogram based bag-of-words or N-gram model which are sparse and can suffer from high dimensionality. There have been later research which explore probabilistic generative models like LDA and pLSA which try to obtain document representations in the topic space instead. These typically result in richer and denser vectors of much fewer dimensions than bag-of-words. Throughout the last decade there have been some attempts at alleviating the network sparsity problem discussed above with the help of text information for all kinds of bibliographic, web and email networks. Some of those methods extend the probabilistic representation of text documents by exploiting their underlying network structure [10,7]. These algorithms show promise as they result in better performances than their content-only (text) or network-only (graph) counterparts on a range of classification tasks. However most of the approaches are semi-supervised and rely on the idea of label propagation throughout the graph and the representations thus created are specific to the task at hand. The notion of injecting textual information specific to an academic citation graph have been studied in [11]. Here the authors make use of potential citations by which they enrich the citation graph and reduce its sparsity. The algorithm proposed for finding these potential citations are based on collaborative filtering and matrix imputation based schemes. A recent approach called TADW[13] was proposed for learning network representations along with text data. To the best of our knowledge this has been the first attempt at tackling the problem of learning fully unsupervised representations of nodes in a graph where the nodes themselves are text data. The learning algorithm in TADW is based on matrix factorization techniques. We treat TADW as an important baseline in our experiments.

There has been a surge of unsupervised feature learning approaches of late which use deep neural networks to learn embeddings in a low dimensional latent vector space. These approaches originated in the field of computer vision and speech signal processing and are now being adopted extensively in other domains. For text, there came shallow neural network based approaches like word2vec [6] and paragraph vectors [5] which are dense word and document representations created using algorithms commonly known as Skip-gram [6]. These approaches are fully unsupervised and are based on the distributional hypothesis "*you shall know a word by the company it keeps*". A flurry of research work in the last few years make use of the so called *word, document embeddings* and achieve state-of-the-art performances throughout the breadth of Natural Language Processing (NLP) tasks [2]. The Skip-gram algorithm introduced in [6] have been extended well beyond words and documents to create representations for nodes in a graph [12,8,3].

We harness the power of these neural networks in our quest of creating rich scientific paper embeddings and propose two novel ways by which we can combine the textual data from papers with the graph information from citation networks. We evaluate Paper2vec against state-of-the-art representations for both graph and text in a multi-class node classification task and a binary link prediction task. Our main contributions in this paper are as follows:

- Introduce Paper2vec - a novel neural network based embedding for representing scientific papers. Propose two novel techniques to incorporate textual information in citation networks to create richer paper embeddings.
- Curate a large collection of almost half a million academic research papers with full text and citation information.
- Conduct experiments on three real world datasets of varied scales and discuss the performance achieved in two evaluation tasks.

The rest of this paper is organized as follows: in Section 2 we discuss the proposed methodology broken into 3 main steps, in Section 3 we give a thorough discussion of our datasets and the experiments conducted. We report our observations and analysis in Section 4 and finally in Section 5 we discuss possible future directions and conclude the paper.

2 The Paper2vec Approach

2.1 Problem Formulation

A citation network dataset can be represented as a graph $G = (V, E)$, where V represents vertices or in our case scientific papers and each edge $e \in E$ represents a citation which links a pair of vertices (v_i, v_j). Neither do we measure strength of citations nor do we differentiate between incoming and outgoing links and thus G is both undirected and unweighted. So we have $(v_i, v_j) \equiv (v_j, v_i)$ and $\omega_{v_i v_j} = 1$. At this point G need not be connected. While in theory, any scientific article is connected to other related works through citation links, in a more real world scenario we can have some papers whose citations are not openly available. In Section 3.1 we conduct our experiments on such a graph.

Let $f : V \rightarrow \mathbb{R}^D$ be the mapping function from the nodes to the representations which we wish to learn. D is the dimensionality of our latent space and $|V|$ is the total number of nodes in our graph (including the non-connected ones). f is a matrix of size $|V| \times D$ parameters. f is learned in two phases by similar optimization algorithms but with different objective functions. The first objective function f_1 aims at capturing the text information while the next f_2 aims at capturing the citation context information. Two ideas are discussed by us which combine information from these different modalities. Note that throughout this paper, we mention the terms vector, embedding or representation. All refer to the same idea of latent vector space for nodes or documents which we aim to learn. We describe each step in detail in the following subsections.

2.2 Phase 1: Learning from Text

As our first step we aim at finding good textual representations for all vertices $v_i \in G$. Since here we consider only the textual information of each paper, we can denote them by a set V $\{d_1, d_2 \ldots d_{|V|}\}$. We use w_i and d_k to denote word and document vectors in our corpus. The original idea for unsupervised learning of document representations [5] was an extension to [6] where we jointly learn document embeddings along with words. First we define a fixed context window $c_1 \in C_1$ over words in a sentence which is slided throughout our corpus. For all the possible contexts in $|C_1|$, we then train every word in a context to predict all words in the context given the document vector and the word vector itself. This results in the document vector contributing to the word prediction task. Effectively we want to maximize the average log probability given in equation 1.

$$\sum_{w_i, w_j \in c_1}^{|C_1|} logPr(w_j|w_i, d_k) \tag{1}$$

where $Pr(w_j|w_i, d_k)$ is defined by the softmax function,

$$Pr(w_j|w_i, d_k) = \frac{exp({w_j}^T w_i + {w_j}^T d_k)}{\sum_{t=1}^{C_1} exp({w_t}^T w_i + {w_t}^T d_k)} \tag{2}$$

The above objective functions can be trained using the Skip-gram algorithm for learning our word and document embeddings. Skip-gram assumes that inside the context, all words are independent of each other and equally important. From Equation 2 we can see that the update step of word vectors w_i require a summation over all the words in our vocabulary which can be huge. We can approximate this objective using positive word-context pairs and randomly sampled negative word-context pairs. An example for negative word-context (w_i, w_j) pair would be *(algorithm, ice-cream)* which is highly unlikely to appear in the same context window. This approach was coined SGNS (skip-gram negative sampling) and it can be used to train our objective function using a 1 hidden layer neural network. Training document embeddings along with words result in a rich representation which is generic in nature and can be further utilized for a variety of domain specific tasks [5]. Among the two architectures (DM, DBOW) proposed in [5], the DM model produced better performance for our evaluation tasks. After this training process, we have all our nodes in a latent space $\mathbb{R}^D$ where textually similar papers are situated closer to each other. We ignore the learned word vectors from here-on and proceed only with $d_{k,\forall k \in V}$.

2.3 Bridging Text Information with Graph

We propose two novel ways of combining the text information into the citation networks. The first method is based on creating artificial text-based links in the citation network. This notion of text-based links can be seen as somewhat analogous to the *potential-citations* introduced in [11]. All vertices in G (papers in our dataset) are already represented as vectors in $\mathbb{R}^D$ as mentioned in Section 2.2. For every node $v_i \in V$ we find the k nearest neighbours of v_i in $\mathbb{R}^D$ and connect them through edges. We call these artificial text-based edges E' and add them to $G' = (V, E \cup E')$. Thus we create links between similar papers in our dataset which are not already linked by citations. Our intuition here is that authors are not always fully aware of all current developments and might miss citing some important papers. In short-papers and poster papers, there is a strict space constraint and authors do not have much choice other than including a mere handful of citations. We claim that the textual similarity between two papers is important and thus aim to bring closer two papers which share similar text content but do not have a direct citation edge between them. In Section 4.1 we provide further details on selecting k with respect to dataset sizes.

Our second method is motivated from the works in the field of computer vision [9], where the authors used pre-trained neural network weights from a general task. They found that their network gave good invariant features when trained on a large dataset that were shown to be generic. Instead of random weight initialization for any domain specific task, they successfully used these pre-trained weights to obtain improved performance across several sub-domain tasks related to computer vision. We use our document vectors learnt from text in Phase 1 as initialization points and further refine them by a new objective function f_2 in Phase 2 which minimizes loss over edges as described in the following subsection.

Later in Figure 3.2 we show through our empirical evaluations that the two aforementioned methods individually contribute to an increase in performance throughout our datasets. We combine both of them to get Paper2vec, a new state-of-the art technique for estimating scientific paper representations.

2.4 Learning from Graph

Henceforward we take G' as our input graph and first define the notion of context or neighbourhood inside G': a valid context $c_2 \in C_2$ for a node v_i is the collection of all nodes v_j that are atmost h hops away from v_i. Value of h is determined by window size of c_2. Note that here we do not differentiate between (v_i, v_j) pairs on whether they are connected by citations or by text-based links. We obtain C_2 by sliding over random walk sequences starting from every node $v_{i,i \in V}$ in G'. Borrowing the same idea from section 2.2, given a node vector v_i we try to predict its neighbouring nodes $v_{j,\forall j \in C_2}$ in the graph. This notion of converting a graph into a series of text documents has been motivated by the fact that word frequency in a document corpus and the visited node frequency during a random walk for a connected graph, both follow the power law distribution [8]. Using the same intuitions as before, we now try to maximize the likelihood function as shown in Equation 3.

$$\sum_{v_i, v_j \in C_2}^{|C_2|} logPr(v_j|v_i) \tag{3}$$

Once more for calculating $Pr(v_j|v_i)$ we can run into the computational problem of summing over all nodes in G' as shown in Equation 4 which can be large. We approximate the objective function by taking sets of positve and negative (v_i, v_j) pairs. In this case, an example for a negative context for v_i would be some vertex v_j which has a very low probability of being in h hop neighbourhood of v_i.

$$Pr(v_j|v_i) = \frac{exp({v_j}^T v_i)}{\sum_{t=1}^{C_2} = exp({v_t}^T v_i)} \tag{4}$$

Applying the CBOW algorithm [6] with negative sampling on our constructed vertex, context pairs would give us our desired representations. This is similar to the strategies discussed by Perozzi et al. in [8] with the difference being in weight initialization.

3 Experimental study

In this section, we begin by discussing the dataset details and our two evaluation metrics. Next we provide a brief overview of all the algorithms we compared against Paper2vec before presenting the performance comparison. We provide discussion and analysis for our chosen methods wherever possible. Towards the end we briefly discuss about hyper-parameter tuning, practical issues, running-time and scalability.

3.1 Datasets

We chose to evaluate Paper2vec on three different academic citation datasets of increasing scale (small, medium, large) described as follows:

- **CORA ML subset**: This is a subset of only Machine Learning papers from the CORA dataset. There are 2,708 papers from 7 classes like *reinforcement learning, probabilistic methods, neural networks* etc. This dataset is connected with 5,429 citation links. For text information we had titles, abstracts and all sentences from a paper containing citations.
- **CORA full dataset**: This is the full CORA dataset containing 51,905 papers from 7 broad categories like *operating systems, databases, information retrieval* etc. We manually pruned out duplicate entities and papers which do not have any associated text information with it. This resulted in a dataset of 36,954 papers with 132,968 citation links within the dataset. We use the same text information as in CORA ML subset.
- **DBLP citation network**(version 2): This dataset is a large collection of computer science papers. DBLP only provides citation-link information and paper titles. For full text of these papers, we refer to a recent research by Zhou et al.[14] which has been crawled from CiteSeerX and is publicly available. This dataset is partly noisy with some duplicate paper information and there is a lack of unique one-to-one mapping from the DBLP paper ids to the actual text of that paper. During the creation of our final dataset, we either pruned out ambiguous papers or manually resolved the conflicts. We came up with a final set of 465,355 papers from the DBLP corpus for which we have full text available. In this set only 224,836 papers are connected by citations because most of the other cited links are outside DBLP (not from computer science domain) and hence full text is not available. However our text-based linking strategy as discussed in Section 2.3 helps us in connecting the graph and getting a final vertex count of 412,806. With only the citations being considered, edge count comes to 2,301,292 (undirected). We gather the required class labels from the MAS dataset [1] by Chakraborty et al. This

dataset contains 807,516 tagged research papers in computer science. Their tags are based on the sub-domains they belong to, in total there are 24 categories like *computer-graphics*, *operating-systems*, *databases*, *language and speech* etc. We took the intersection of these labels with our cleaned DBLP dataset and found 134,338 matches for our classification experiment.

3.2 Comparison to Previous Work:

For comparison with Paper2vec first we chose both text-only and network-only algorithms. Along with this we compared against a recently proposed text and graph combined algorithm and a concatenated baseline method for combining text and graph representations. The algorithms are discussed below:

– **Deepwalk**[8] is a network-only algorithm which represents a graph as a series of text streams and learns the node representations by applying the SGNS algorithm.
– **TADW**[13] or Text Associated DeepWalk is a matrix factorization based approach to approximate Deepwalk. It also uses *tf-idf* features to fuse link and text data similar to ours.
– **LINE**[12] learns network-only graph representations in two phases - first order proximity and second order proximity. Their edge sampling algorithm is similar to our discussed negative sampling.
– **Paragraph Vector**[5] is the original algorithm proposed by Le et al. for learning latent representation for text documents. We use this algorithm in our text learning step to get pre-trained vectors. This serves as a text-only (content-only) baseline.
– **tf-idf**[4] is an improvement over the simple bag-of-words algorithm for representing documents in the vector space.
– **Concatenated baseline**: We concatenated Paragraph Vector with Deepwalk embeddings to serve as a baseline for our text-graph combination.

3.3 Evaluation Tasks

We chose two tasks to evaluate our learned embeddings. Across all our datasets, we thus conduct 6 sets of experiments to demonstrate the effectiveness of Paper2vec embeddings.

Node classification: In this task we need to determine the class or label of a scientific paper given its representation. For the text-only methods, this problem can be treated as multi-class document classification. After the (unsupervised) feature learning phase from respective algorithms we evaluate the classification accuracy across our three datasets. We vary the training ratio from 10% to 50% (rest are treated as test-set) and report the scores for each trial averaged over 10 times. This is the exact experimental details found in [13].

Link prediction: Here we are given a random pair of node representations (v_i, v_j) and we need to determine whether there should be a citation link between them. For every pair we have two representations - one for each node. We use the

Hadamard operator to combine two node vectors into one edge representation. Our edge representations $f_2(E_{i,j}) = f_2(v_1) * f_2(v_2)$ remain in $\mathbb{R}^D$ as this operator performs element wise multiplication between the node vectors. Grover et al. in [3] studied in detail this problem of edge (citation) representation by combining dense node embeddings. We remove 25% of citation links from each of our three datasets before starting the representation training phase. For creating the link prediction evaluation dataset, we have this 25% removed citation links (positive examples) and we add random links to each graph which were not originally present (negative examples). Now we have a binary link prediction problem where given two nodes, we need to predict the presence of a link between them. We took care not to mix our text-based links with this random negative samples. Our final link-prediction dataset contains 20,000, 60,000 and 440,000 examples for the small, medium and large datasets respectively. We report scores for 5-fold cross-validation on this binary classification task for all algorithms.

Table 1: Node Classification performance on CORA (ML-subset) dataset.

Type of embedding	Algorithm (dimensions)	SVM training ratio				
		10%	*20%*	*30%*	*40%*	*50%*
network-only	LINE (100)	68.32	71.68	74.92	78.06	79.95
	Deepwalk (100)	75.96	80.66	82.71	84.31	85.30
content-only	Paragraph Vector (100)	76.83	81.12	82.82	83.54	84.41
	tf-idf (1433)	78.48	82.83	84.73	85.99	86.88
combined	Concatenated baseline (200)	80.61	83.76	85.13	86.35	87.14
	TADW (160)	82.4	85.0	85.6	86.0	86.7
	Paper2vec (100)	**83.41**	**86.49**	**87.46**	**88.26**	**88.85**

Table 2: Node Classification performance on CORA (full) dataset.

Type of embedding	Algorithms (dimensions)	SVM training ratio				
		10%	*20%*	*30%*	*40%*	*50%*
network-only	LINE (200)	76.98	79.35	80.46	81.34	81.82
	Deepwalk (200)	77.97	80.24	81.29	82.17	82.65
content-only	Paragraph Vector (200)	75.46	78.34	79.54	80.40	80.87
	tf-idf (5000)	76.29	77.21	78.24	79.14	80.34
combined	Concatenated baseline (500)	80.16	81.72	82.68	83.37	83.88
	TADW (200)	77.51	79.69	80.68	81.17	81.36
	Paper2vec (200)	**82.31**	**83.83**	**84.45**	**84.87**	**85.18**

3.4 Classifier Details

We use Support Vector Machines(SVM) similar to those in [13] for all our node classification tasks. In the link prediction tasks we present results with Logistic Regression as they performed better than SVM. We use the Python library *scikit-learn* for our classifier implementations. Since we aim at comparing the learned embeddings, we focus less on exact classifier settings. However we only report the best score achieved by every algorithm in each dataset.

Table 3: Node Classification performance on DBLP dataset. TADW scores are unavailable due to scalability issues.

Type of embedding	Algorithms (dimensions)	SVM training ratio				
		10%	*20%*	*30%*	*40%*	*50%*
network-only	LINE (500)	61.36	62.38	62.74	63.11	63.61
	Deepwalk (500)	60.27	61.56	62.29	63.00	63.43
content-only	Paragraph Vector (500)	57.26	58.03	59.35	60.12	60.45
	td-idf (50000)	57.57	58.59	58.87	59.5	60.09
combined	Concatenated baseline (800)	64.56	65.78	66.35	66.98	67.77
	Paper2vec (500)	**65.45**	**66.46**	**67.61**	**68.21**	**69.94**

Table 4: Link Prediction performance on all datasets. Presented scores are *micro-f1* with 5-fold cross-validation. All dimensions are kept same as node classification task with respect to dataset.

Type	Algorithm	CORA(ML)	CORA(full)	DBLP
network-only	LINE	79.22	91.38	95.08
	Deepwalk	81.30	92.70	94.92
content-only	Paragraph Vector	77.57	85.59	88.87
combined	Concatenated baseline	83.13	93.36	95.32
	TADW	85.65	81.67	--
	Paper2vec	**91.75**	**95.11**	**97.13**

4 Results and Discussion

We set the dimension k in TADW at 160 as recommended by the authors in [13] for our CORA ML (small) dataset. In the CORA full (medium) dataset, we see an improvement in performance on increasing $k = 200$. For Deepwalk on all our datasets and evaluation tasks we found the number of walks started per vertex $\gamma = 10$ and window size $t = 5$ to perform best. Note that this is contradicting with the values set for Deepwalk ($\gamma = 80, t = 10$) in [13] and thus the results vary. We keep the dimensions of LINE, Deepwalk and Paragraph Vectors exactly the same as Paper2vec: 100 for small, 200 for medium and 500 for large. During our evaluation for *tf-idf* text based baselines we kept the maximum features (sorted by *df* value) as 1433 for our small [13] and 5000 for our medium dataset. We list below our major observations:

– Paper2vec is able to out-perform all baselines and competing algorithms throughout all datasets in both the tasks. It can be inferred from Figure 1 that both our text infusion methods with graphs, individually increase the performance over baseline network-only methods.
– From our results in Tables 1 and 2, we can see that neural network based techniques are generally able to out-perform the matrix factorisation based methods. Similar trends have also been reported in [8,12].

– Our chosen concatenated baseline method of concatenating Deepwalk embeddings with Paragraph Vector is consistently able to outperform both individually and also TADW, though at the cost of (double) dimension size.
– Surprisingly Deepwalk embeddings are quite competitive with the content-only algorithms in our node classification task. This is quite astonishing given that unlike Paragraph Vector, Deepwalk was trained with no information about text. A behavior like this can be attributed to our domain of scientific data where authors tend to cite more papers from the same domain. LINE is able to perform better on the large DBLP dataset.
– Content-only algorithms resulted with better scores in our node classification task as expected. Their performance in link prediction fall short of the graph and combined embedding algorithms. It is interesting to observe that during node classification, *tf-idf* features perform quite well on the small and medium datasets. It's performance reaches the level of the combined method TADW. Results reported here for *tf-idf* are better than the binary feature based *bag-of-words* approach in [13].
– In the link prediction task, we can see on the CORA-ML (small) dataset, there is an improvement of above 11% for Paper2vec over Deepwalk. As we grow our dataset sizes, more citation links make the graph denser thus diminishing the effect of text information for this task.
– Our unsupervised representations are able to surpass prior work based on semi-supervised models like Collective Classification [10] in the first task.

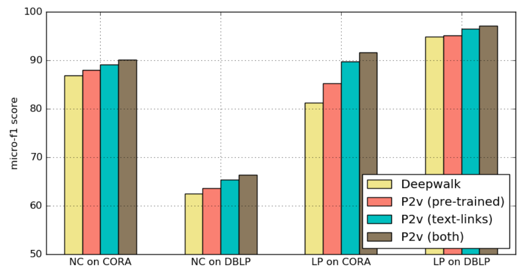
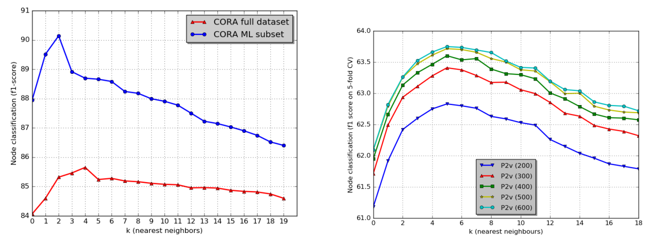


Fig. 1: Comparison of our text-infusion methods against baseline (Deepwalk). Here P2v refers to Paper2vec. NC and LP denotes node classification and link prediction respectively. The scores are based on 5-fold cross-validation.

Through these six experiments we show that in Paper2vec, we are able to successfully fuse text with graph without any trade-offs or loss of information. Our representations in $\mathbb{R}^D$ are able to perform better than concatenated features in $\mathbb{R}^{2D}$ (baseline), previous state-of-the-art in link-text combination (TADW), all text-based methods for node classification and graph-based methods for link prediction. These results signify the vital role which textual data plays in creating rich embeddings for scientific papers.



(a) CORA ML and full datasets (b) DBLP dataset

Fig. 2: Node classification performance with varying k and D for Paper2vec.

4.1 Parameter Sensitivity

We used the DM model [5] and CBOW model [6] respectively for our text and graph learning stages. Varying window size of c_1 in text learning framework from 5 to 10 did not have notable impact on our document vectors, we fixed it at 10. However c_2, window size for the graph learning framework gave best results for the value of 5 on being varied from 2 to 10. This signifies that our paper similarity is best determined by its 5-hop neighbourhood distribution. During DM training for optimising f_1 we ran 10 epochs by setting a learning-rate decay of 0.02 after every epoch and keeping it constant throughout the epoch. The two remaining hyper-parameters to tune for Paper2vec are number of neighbors to connect k and our embedding dimensions D. From Figures 2a and 2b we can see a steady increase of node classification performance as we connect $2, 4, 5$ neighbours for our small, medium and large datasets respectively. After this performance peak, there is a steady decline as we start connecting arbitrary (less similar) nodes together. In this experiment D was kept constant at 100 and 200 for small and medium datasets respectively. Similarly in Figure 2b we see a constant increase in performance by cranking up D on our large dataset. Bigger networks contain more data and we need to keep a higher value for D to capture all of it. However beyond a certain limit ($D = 500$ for DBLP), this gain diminishes. We used the popular library gensim for both optimisation (f_1, f_2) implementations.

4.2 Runtime and Scalability

We conducted all our experiments on a single desktop PC with specifications: Intel Pentium G3220 processor and 8GB memory. Every neural network based algorithm (Deepwalk, LINE, Paragraph Vector) including ours, were scalable to handle the DBLP dataset. However for our text-only baseline *tf-idf* we had to run mini-batch stochastic gradient descent for the classification task due to memory limitations. Unfortunately for the TADW algorithm, its matrix factorization based approach was not directly scalable to our DBLP dataset.

5 Conclusion and Future Work

In this paper, we present Paper2vec, a novel algorithmic framework for unsupervised learning of combined textual and graph features in academic citation networks. Our algorithm shows two ways by which we can incorporate the learned text representations with graph features and achieve a higher overall predictive performance in various tasks. For both our node classification and link prediction tasks, we find Paper2vec perform superior to state-of-the-art techniques across all datasets. A future research direction would be to devise an inference procedure for unseen papers not present within the training corpus. Through this we intend to explore the dynamics of our system in a more real world scenario where the incoming number of new papers is quite high and how we can incorporate them inside our model without loss of prior information. In addition to this, we hope to generalize our algorithm further and expand it to graphs beyond academic literature. Most of the web-data available today are connected by hyper-links and contain text information. We look forward to exploit information from different modalities and create rich representations across domains.

References

1. Chakraborty, T., Sikdar, S., Tammana, V., Ganguly, N.: Computer science fields as ground-truth communities: their impact, rise and fall. In: ASUNAM. (2013)
2. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.P.: Natural language processing (almost) from scratch. CoRR **abs/1103.0398** (2011)
3. Grover, A., Leskovec, J.: Scalable feature learning for networks. In: KDD. (2016)
4. Joachims, T.: A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, DTIC Document (1996)
5. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. In: ICML. (2014)
6. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. CoRR (2013)
7. Nallapati, R., Cohen, W.W.: Link-plsa-lda: A new unsupervised model for topics and influence of blogs. In: AAAI. (2008)
8. Perozzi, B., Al-Rfou', R., Skiena, S.: Deepwalk: online learning of social representations. In: KDD. (2014)
9. Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: Cnn features off-the-shelf: An astounding baseline for recognition. In: CVPR. (2014)
10. Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine **29** (2008)
11. Sugiyama, K., Kan, M.Y.: Exploiting potential citation papers in scholarly paper recommendation. In: JCDL. (2013)
12. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: Large-scale information network embedding. In: WWW, ACM (2015)
13. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.Y.: Network representation learning with rich text information. In: IJCAI. (2015)
14. Zhou, T., Zhang, Y., Lu, J.: Classifying computer science papers. In: IJCAI. (2016)