Linked Document Embedding for Classification

Suhang Wang[†], Jiliang Tang[‡], Charu Aggarwal[#], and Huan Liu[†]
[†]Computer Science & Engineering, Arizona State University, Tempe, AZ, USA
[‡]Computer Science & Engineering, Michigan State University, East Lansing, MI, USA
[#]IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
[†]{suhang.wang,huan.liu}@asu.edu, [‡]jiliang.tang@cse.msu.edu, [#]charu@us.ibm.com

ABSTRACT

Word and document embedding algorithms such as Skip-gram and Paragraph Vector have been proven to help various text analysis tasks such as document classification, document clustering and information retrieval. The vast majority of these algorithms are designed to work with independent and identically distributed documents. However, in many real-world applications, documents are inherently linked. For example, web documents such as blogs and online news often have hyperlinks to other web documents, and scientific articles usually cite other articles. Linked documents present new challenges to traditional document embedding algorithms. In addition, most existing document embedding algorithms are unsupervised and their learned representations may not be optimal for classification when labeling information is available. In this paper, we study the problem of linked document embedding for classification and propose a linked document embedding framework LDE, which combines link and label information with content information to learn document representations for classification. Experimental results on real-world datasets demonstrate the effectiveness of the proposed framework. Further experiments are conducted to understand the importance of link and label information in the proposed framework LDE.

Keywords

Document Embedding, Linked Data, Word Embedding

1. INTRODUCTION

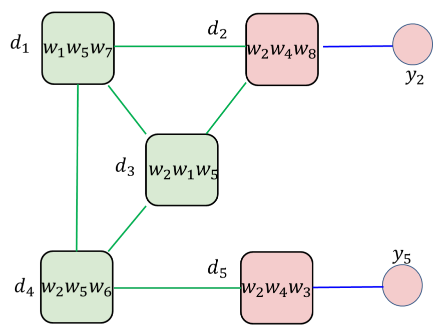
A meaningful and discriminative representation for documents can help many text analysis tasks such as document classification, document clustering and information retrieval. Many document representation methods are proposed such as bag-of-words, N-gram, latent semantic analysis [12], latent Dirichlet allocation [5] and word/document embedding [18, 17, 13]. Among these algorithms, the recently proposed distributed representations of words and documents such as Skip-gram [18, 17] and PV-DM [13] have demonstrated superior performance in many tasks such as word analogy [18], parsing [9], POS tagging [9], and sentiment analysis [11]. The assumption behind these document/word embedding approaches is basically the distributional hypothesis that "you shall know a word by the company it keeps [10]." They embed words or documents into a low dimensional space, which can alleviate the curse of dimensionality and data sparsity problems suffered by traditional representations such as bag-of-words and N-gram.

The vast majority of existing document embedding algorithms work with "flat" data and documents are usually assumed to be independent and identically distributed (or *i.i.d.* assumption). However, in many real-world scenarios, documents are inherently linked. For example, web documents such as blogs and online news often contain hyperlinks to other web documents, and scientific articles commonly cite other articles. A toy example of linked documents is illustrated in Figure 1 where $\{d_1, d_2, \ldots, d_5\}$ are documents and $\{w_1, w_2, \ldots, w_8\}$ are words in documents. In addition to content information, documents are linked and links suggest the inter-dependence of documents. Hence, the *i.i.d.* assumption of documents does not hold [33]. Additional link information of such documents has been shown to be useful in various text mining tasks such as document classification [33, 8], document clustering [14, 29] and feature selection [27]. Therefore, we propose to study the novel problem of linked document embedding following the distributional hypothesis.

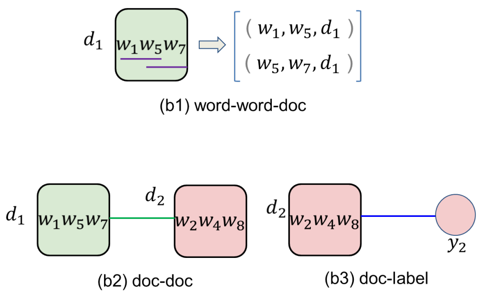
Most existing document embedding algorithms use unsupervised learning, such as those in [18, 13, 32]. The representations learned by these algorithms are very general and can be applied to various tasks. However, they may not be optimal for some specialized tasks where label information is available such as y_2 for d_2 and y_5 for d_5 in Figure 1(a). For example, deep learning algorithms such as convolutional neural networks [11], which use label information, often outperform text embeddings for classification tasks [23]. Hence, in this paper we study the novel problem of linked document embedding for classification and investigate two specific problems: (1) how to capture link and label information mathematically; and (2) how to exploit them for document embedding. In an attempt to address these two problems, we propose a novel linked document embedding (LDE) framework for classification. The major contributions of the paper are summarized next:

- We provide a principled way to capture link and label information mathematically;
- We propose a novel framework LDE, which learns word

CIKM'16 , October 24-28, 2016, Indianapolis, IN, USA
© 2016 ACM. ISBN 978-1-4503-4073-1/16/10. . . $15.00
DOI: http://dx.doi.org/10.1145/2983323.2983755



(a) Linked Documents



(b) Three Types of Relations

Figure 1: A Toy Example of Linked Documents. $\{d_1, d_2, \ldots, d_5\}$ **are documents;** $\{w_1, w_2, \ldots, w_8\}$ **are words;** y_2 **is the label of** d_2 **and** y_5 **is the label of** d_5**.**

and document embeddings for classification by combining link and label information with content information; and

- We conduct experiments on real-world datasets to understand the effectiveness of the proposed framework LDE.

The rest of the paper is organized as follows. In Section 2, we briefly review the related work. In Section 3, we formally define the problem of linked document embedding for classification. In Section 4, we introduce the proposed framework LDE with the details about how to model link and label information and how to incorporate them in document and word embedding. In Section 5, we show how to solve the optimization problem of LDE along with solutions to accelerate the learning process. In Section 6, we present empirical evaluation with discussion. In Section 7, we conclude with future work.

2. RELATED WORK

In this paper, we investigate linked document representation for classification, which is mainly related to document representation, linked data and graph-based classification.

2.1 Document Representation

Document representation is an important research area that receives great attention lately and can benefit many machine learning and data mining tasks such as document classification [23], information retrieval [31, 20] and sentiment analysis [13]. Many different types of models have been proposed for document representation. Bog-of-words [21] is one of the most widely used one. It is simple to implement, but not scalable since as the number of documents increases, the vocabulary size can become huge. At the same time, it suffers from data sparsity and curse of dimensionality problems and the semantic relatedness between different words is omitted. To mitigate the high dimensionality and data sparsity problems of BOW, Latent Semantic Analysis [12] uses dimensionality reduction technique, i.e., SVD, to project the document-word matrix to a low dimension space. Latent Dirichlet Allocation [5] is another low dimensional document representation algorithm. It is a generative model that assumes that each document has topic distribution and each word in the document is drawn from a topic with probability. Recently, Mikolov et al. proposed the distributed representations of words, Skip-gram and CBOW, which learn the embeddings of words by utilizing word co-occurrence in the local context [18, 17]. It has been proven to be powerful to capture the semantic and syntactic meanings of words and can benefit many natural language processing tasks such as word analogy [18], parsing [9], POS tagging [9], and sentiment analysis [11]. It is also scalable and can handle millions of documents. Based on the same distributed representation idea, [13] extended the word embedding model to document embedding (PV-DM, PV-DBOW) by finding document representations that are good at predicting words in the document. Document embedding has also been proven to be powerful in many tasks such as sentiment analysis [13], machine translation [28] and information retrieve [20]. Recently, predictive text embedding algorithm (PTE) is proposed in [23], which also utilizes label information to learn predictive text embeddings. The proposed framework LDE is inherently different from PTE: (1) LDE is developed for linked documents while PTE still assumes documents to be *i.i.d.*; (2) LDE captures label information via modeling document-label information while PTE uses label information via word-label information; (3) in addition to label information, LDE also models link information among documents to learn document embeddings; and (4) the proposed formulations and optimization problems of LDE are also different from those of PTE.

2.2 Linked Document Representation

Documents in many real-world applications are inherently linked. For example, web pages are linked by hyperlinks and scientific papers are linked by citations. Link information has been proven to be very effective for machine learning and data mining such as feature selection [26, 27], recommender systems [15, 16], and document classification/clustering [19, 3]. Based on the idea that two linked documents are likely to share similar topics, several works have been proposed to utilize link information for better document representations [7, 35, 32]. For example, RTM [7] extends LDA by considering link information for topic modeling; PMTLM [35] combines topic modeling with a variant of mixed-membership block model to model linked documents and TADW [32] learns linked document representations based on matrix factorization. However, the majority of the aforementioned works do not utilize label information; meanwhile most of them do not learn distributed document representations based on the distributional hypothesis; while LDE employs distributional hypothesis idea for document embedding by combining link and label information with content simultaneously.

2.3 Graph-based Classification

Graph-based classification is to utilize the link information to design classifier for classification. Various graph-based classification algorithms are proposed. Label propagation [34] is a classical graph-based methods, which performs classification by propagating label information from labeled data to unlabeled data through the graph. However, label propagation doesn't utilize the features of documents. GC [4] is a more advanced graph-based classification method, which takes into account both link structure and documents' content and can be combined with SVM classifiers. Graffiti [2] is proposed to perform random walk on heterogeneous networks so as to capture the mutual influence of connected nodes for classification. Abernethy et al. [1] incorporates the graph information into SVM classifier for web spam detection. The proposed method LDE is inherently different form the existing graph-based classification. First, LDE learns both word embedding and document embedding, which can be used for other tasks, such as word analogy [18] and visualization [23]; while existing graph-based classification methods don't learn word/document representation. Second, LDE utilizes the distributional hypothesis idea and considers word-word-document relations, while existing graph-based classification methods usually use BOW without considering the word-word relationship.

3. PROBLEM STATEMENT

We first introduce notations used in this paper. Throughout the paper, matrices are written as boldface capital letters and vectors are denoted as boldface lowercase letters. For an arbitrary matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$, $\mathbf{M}_{ij}$ is the (i,j)-th entry of $\mathbf{M}$ while $\mathbf{m}^i$ and $\mathbf{m}_j$ are the i-th row and j-th column of $\mathbf{M}$, respectively. $\|\mathbf{M}\|_F$ is the Frobenius norm of $\mathbf{M}$. Capital letters in calligraphic math font such as $\mathcal{P}$ are used to denote sets. $|\mathcal{P}|$ is the cardinality of a set $\mathcal{P}$.

Let $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$ be a set of N documents and $\mathcal{W} = \{w_1, w_2, \ldots, w_M\}$ be the word dictionary of size M for $\mathcal{D}$. Documents in $\mathcal{D}$ are linked, which forms a document network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each vertex is a document and $e_{ij} = 1$ if documents d_i and d_j are connected. We use $\mathcal{Y}$ to denote the subset of labeled documents in $\mathcal{D}$ where y_i represents label information of the document d_i. Let $\mathbf{D} \in \mathbb{R}^{d \times N}$ be the document embedding matrix where the i-th column of $\mathbf{D}$, i.e., $\mathbf{d}_i \in \mathbb{R}^{d \times 1}$, is a d-dimensional vector representation of the document d_i. Similarly we use $\mathbf{W} \in \mathbb{R}^{d \times M}$ to denote the word embedding matrix where the j-th column of $\mathbf{W}$, i.e., $\mathbf{w}_j \in \mathbb{R}^{d \times 1}$, is a d-dimensional vector representation of the word w_j in $\mathcal{W}$.

With aforementioned definitions and notations, the problem under study is formally stated as:

Given the document set $\mathcal{D}$, the document network $\mathcal{G}$ and partial label information of $\mathcal{D}$, i.e., $\mathcal{Y}$, we want to learn the document embedding matrix $\mathbf{D}$ and the word embedding matrix $\mathbf{W}$. Mathematically, the problem is written as :

$$f(\mathcal{D}, \mathcal{G}, \mathcal{Y}) \rightarrow \{\mathbf{D}, \mathbf{W}\} \tag{1}$$

where f is the learning algorithm we propose to investigate.

4. THE PROPOSED FRAMEWORK

To model content, link and label information for word and document embedding, we extract three types of relations by examining Figure 1(a). The three types of relations are demonstrated in Figure 1(b): (1) word-word-document relations from content information shown in Figure 1(b1); (2) document-document relations from link information shown in Figure 1(b2); and (3) document-label relations from label information shown in Figure 1(b3). Next we elaborate these three relations and their corresponding model components before introducing the proposed framework LDE.

4.1 Modeling Word-Word-Document Relations

The distributional hypothesis that "you shall know a word by the company it keeps" suggests that a word has close relationships with its neighboring words. For example, the phrases *win the game* and *win the lottery* appear very frequently; thus the pair of words *win* and *game* and the pair of words *win* and *lottery* could have very close relationship. When we are only given the word *win*, we would highly expect the neighboring words to be words like *game* or *lottery* instead of words as *light* or *air*. This suggests that a good word representation should be useful for predicting its neighboring words, which is the essential idea of Skip-gram [18]. Meanwhile, depending on the topics of the documents, the probabilities of words appearing in the documents are different [5]. For example, though the appearance of the phrase *win the lottery* is frequent, if we know that the topic of a document is about "sports", we would expect words as *game* or *competition* after the word *win* instead of the word *lottery* because *win the game/competition* is more reasonable under the topic of "sports". On the contrary, if the topic of the documents is about "lottery", then we would expect *lottery* after *win*. These intuitions suggest that the predictions of neighboring words for a given word also strongly rely on the document. Therefore, we extract word-word-document relations from content information.

For a word w_i, we use a window of size c to extract w_i and its $(c-1)$ neighbors with w_i at the center and then w_i and each of its $c-1$ neighbors w_j form a pair as (w_i, w_j). At the same time, we record which document the pair of words (w_i, w_j) comes from, say d_k. The pair of words (w_i, w_j) and the document d_k form a triplet (w_i, w_j, d_k). An illustrative example of such process is given in Figure 1(b1), where window size c is 2. We denote all these triplets as a set $\mathcal{P}$. Note that in $\mathcal{P}$, there may be multiple (w_i, w_j, d_k) if the co-occurrence of w_i and w_j happens multiple times in d_k and there may be also (w_i, w_j, d_s) and (w_i, w_j, d_k) if the co-occurrence of w_i and w_j appears in both d_s and d_k. After extracting $\mathcal{P}$, the word-word-document relations can be captured by maximizing the average log probability:

$$\max_{\mathbf{W},\mathbf{D}} \frac{1}{|\mathcal{P}|} \sum_{(w_i, w_j, d_k) \in \mathcal{P}} \log P(w_j | w_i, d_k) \tag{2}$$

where $P(w_j|w_i, d_k)$ means the probability of given d_k, word w_j is a neighboring word of w_i, which is defined as

$$P(w_j|w_i, d_k) = \frac{\exp(\mathbf{w}_j^T \mathbf{w}_i + \mathbf{w}_j^T \mathbf{d}_k)}{\sum_{t=1}^{M} \exp(\mathbf{w}_t^T \mathbf{w}_i + \mathbf{w}_t^T \mathbf{d}_k)} \tag{3}$$

4.2 Modeling Document-Document Relations

Links between documents indicate the inter-dependence of documents. For example, a piece of online news about "sports" is likely to have hyperlinks to other news on "sports" and a web mining article is likely to cite other web mining articles. Two linked documents are likely to share similar topics, which is a property commonly exploited in many tasks such as classification [22] and feature selection [26]. Therefore, we extract document-document relations from link information. For two linked document d_i and d_j, i.e., $e_{ij} = 1$, the embedding vector for d_i is a good indicator of that of d_j since they are likely to share similar topics, which can be achieved by the following optimization problem:

$$\max_{\mathbf{D}} \frac{1}{|\mathcal{E}|} \sum_{i=1}^{N} \sum_{j:e_{ij}=1} \log P(d_j|d_i) \tag{4}$$

where $|\mathcal{E}|$ is the number of links and $P(d_j|d_i)$ is given as

$$P(d_j|d_i) = \frac{\exp(\mathbf{d}_j^T \mathbf{d}_i)}{\sum_{k=1}^{N} \exp(\mathbf{d}_k^T \mathbf{d}_i)} \tag{5}$$

From Eq.(5), we can see that if two linked documents have similar representations, then $P(d_j|d_i)$ will be large. Thus, Eq.(4) aims at maximizing the similarity between two linked documents based on their embedding vectors.

4.3 Modeling Document-Label Relations

For the classification problem, we have some labeled samples and label information could guide the document embedding algorithms to learn better embeddings. Let $\mathbf{Y} \in \mathbb{R}^{d \times N_c}$ be the label embedding matrix where N_c is the number of unique labels and the k-th column of $\mathbf{Y}$, $\mathbf{y}_k$, is the embedding vector for the k-th label. y_i is the label of the i-th document and the corresponding label embedding vector for y_i is $\mathbf{y}_{y_i}$. However, to avoid the notation confusion, we use $\mathbf{y}_{d_i}$ instead of $\mathbf{y}_{y_i}$ to denote the representation of the class label that is assigned to d_i in the remainder of the paper. A good document embedding vector for d_i should be a good indicator of the label of d_i. In other words, given the document, we should be able to predict its label; hence we extract document-label relations from label information, which can be captured as follows:

$$\max_{\mathbf{Y},\mathbf{D}} \frac{1}{|\mathcal{Y}|} \sum_{i:y_i \in \mathcal{Y}} \log P(y_{di}|d_i) \tag{6}$$

where $P(y_{d_i}|d_i)$ is the probability that d_i's label is y_{d_i}, which is given as

$$P(y_{d_i}|d_i) = \frac{\exp(\mathbf{y}_{d_i}^T \mathbf{d}_i)}{\sum_{k=1}^{N_c} \exp(\mathbf{y}_k^T \mathbf{d}_i)} \tag{7}$$

4.4 Linked Document Embedding

With model components to capture content, link and label information, the proposed linked document embedding framework LDE is to solve the following optimization problem:

$$\begin{aligned}\min_{\mathbf{W},\mathbf{D},\mathbf{Y}} & -\frac{1}{|\mathcal{P}|} \sum_{(w_i,w_j,d_k)\in\mathcal{P}} \log P(w_j|w_i,d_k) \\ & -\frac{1}{|\mathcal{E}|} \sum_{i=1}^{N} \sum_{j:e_{ij}=1} \log P(d_j|d_i) \\ & -\frac{1}{|\mathcal{Y}|} \sum_{i:y_i\in\mathcal{Y}} \log P(y_i|d_i) + \gamma\Omega(\mathbf{W},\mathbf{D},\mathbf{Y})\end{aligned} \tag{8}$$

In Eq.(8), the first term aims to learn document and word embeddings that are useful for predicting the neighbor word, which captures content information. The second term models link information and the third term captures label information that allows the document embeddings with the capability to predict labels. $\Omega(\mathbf{W},\mathbf{D},\mathbf{Y})$ is the regularizer to prevent the model from overfitting. We can choose the regularizer with ℓ_1-norm if the dimension of the embedding is high. Since we want to represent documents and words with low-dimensional vectors for classification, we use the Frobenius in our model:

$$\Omega(\mathbf{W},\mathbf{D},\mathbf{Y}) = \|\mathbf{W}\|_F^2 + \|\mathbf{D}\|_F^2 + \|\mathbf{Y}\|_F^2 \tag{9}$$

5. LEARNING LDE

In this section, we introduce the details of how to use stochastic gradient method to train the model. We will first introduce how to speed up the training process and then give the update rules and the detailed algorithm.

5.1 Approximation by Negative Sampling

To update $\mathbf{w}_j$, we need to take derivative of $\log P(w_j|w_i,d_k)$ w.r.t. $\mathbf{w}_j$, which is given by:

$$\nabla_{\mathbf{w}_j} \log P(w_j|w_i,d_k) = (1 + P(w_j|w_i,d_k))(\mathbf{w}_i + \mathbf{d}_k) \tag{10}$$

From Eq.(10), we find that updating $\mathbf{w}_j$ requires the calculation of $P(w_j|w_i,d_k)$. However, the calculation of $P(w_j|w_i,d_k)$ is expensive, because the denominator of $P(w_j|w_i,d_k)$ is written as $\sum_{t=1}^{M} \exp(\mathbf{w}_t^T\mathbf{w}_i + \mathbf{w}_t^T\mathbf{d}_k)$. It requires summation over all the words, which could be very inefficient since the number of words is usually very large. To accelerate the speed, following the method used in Skip-gram model, we use the trick of negative sampling. In detail, the negative sampling is defined by the following objective function [18]:

$$\log \sigma(\mathbf{w}_j^T(\mathbf{w}_i + \mathbf{d}_k)) + \sum_{t=1}^{K} \mathbb{E}_{w_t \sim P_n(w)}[\log \sigma(-\mathbf{w}_t^T(\mathbf{w}_i + \mathbf{d}_k))] \tag{11}$$

which replaces every $\log P(w_j|w_i,d_k)$ term in the objective function of Eq.(8). Thus the task becomes to distinguish the target word w_j from K words drawn from the noise distribution $P_n(w)$. The idea behind negative sampling is that we want to maximize the similarity between $\mathbf{w}_j$ and $(\mathbf{w}_i + \mathbf{d}_k)$ and minimize the similarity between a randomly sampled word $\mathbf{w}_t$ and $(\mathbf{w}_i + \mathbf{d}_k)$. In this way, we can approximately maximize $\log P(w_j|w_i,d_k)$. In practice, the noise distribution is chosen to be $U(w)^{3/4}/Z$, where $U(w)$ is the unigram distribution of the words and $Z = \sum_w U(w)^{3/4}$ is the normalization term.

Thus, for a training instance $(w_i,w_j,d_k) \in \mathcal{P}$, we would draw K negative word samples, say one negative sample is w_t, from the noise distribution as $w_t \sim P_n(w)$ and then we put (w_i,w_t,d_k) into the negative training set $\mathcal{N}$. It is easy to verify that $|\mathcal{N}| = K|\mathcal{P}|$. Now with $\mathcal{N}$ and $\mathcal{P}$, we can approximate the first term in Eq.(8) using Eq.(11) as

$$\begin{aligned}\min_{\mathbf{W},\mathbf{D}} & -\frac{1}{|\mathcal{P}|} \sum_{(w_i,w_j,d_k)\in\mathcal{P}} \log \sigma(\mathbf{w}_j^T(\mathbf{w}_i + \mathbf{d}_k)) \\ & -\frac{1}{|\mathcal{P}|} \sum_{(w_i,w_t,d_k)\in\mathcal{N}} \log \sigma(-\mathbf{w}_t^T(\mathbf{w}_i + \mathbf{d}_k))\end{aligned} \tag{12}$$

Similarly, we approximate $P(d_j|d_i)$ as

$$\log \sigma(\mathbf{d}_j^T\mathbf{d}_i) + \sum_{t=1}^{K} \mathbb{E}_{d_t \sim \tilde{P}_n(d)} \log \sigma(-\mathbf{d}_t^T\mathbf{d}_i) \tag{13}$$

where d_t in Eq.(13) is randomly sampled from documents that are not linked with d_i. Thus, for each linked document pair (d_i, d_j), we need to randomly sample K documents, d_t, that are not linked to d_i and put (d_i, d_t) to the negative document set $\mathcal{N}_E$. We can see that $|\mathcal{N}_E| = K|\mathcal{E}|$. Now the second term in Eq.(8) can be written as

$$\min_{\mathbf{D}} -\frac{1}{|\mathcal{E}|}\left(\sum_{e_{ij}\in\mathcal{E}} \log\sigma(\mathbf{d}_j^T\mathbf{d}_i) + \sum_{(d_i,d_t)\in\mathcal{N}_E} \log\sigma(-\mathbf{d}_t^T\mathbf{d}_i)\right) \tag{14}$$

With the similar idea, $P(y_{d_i}|d_i)$ can be approximated as

$$\log\sigma(\mathbf{y}_{d_i}^T\mathbf{d}_i) + \sum_{y\neq y_{d_i}} \log\sigma(-\mathbf{y}_y^T\mathbf{d}_i) \tag{15}$$

With these negative sampling approximations, the objective function of Eq.(8) can be approximated as

$$\begin{aligned}\min_{\mathbf{W},\mathbf{D},\mathbf{Y}} & -\frac{1}{|\mathcal{P}|}\sum_{(w_i,w_j,d_k)\in\mathcal{P}} \log\sigma(\mathbf{w}_j^T(\mathbf{w}_i+\mathbf{d}_k)) \qquad (16)\\ & -\frac{1}{|\mathcal{P}|}\sum_{(w_i,w_t,d_k)\in\mathcal{N}} \log\sigma(-\mathbf{w}_t^T(\mathbf{w}_i+\mathbf{d}_k))\\ & -\frac{1}{|\mathcal{E}|}\sum_{e_{ij}\in\mathcal{E}}\log\sigma(\mathbf{d}_j^T\mathbf{d}_i) - \frac{1}{|\mathcal{E}|}\sum_{(d_i,d_t)\in\mathcal{N}_E}\log\sigma(-\mathbf{d}_t^T\mathbf{d}_i)\\ & -\frac{1}{|\mathcal{Y}|}\sum_{i:y_{d_i}\in\mathcal{Y}}[\log\sigma(\mathbf{y}_{d_i}^T\mathbf{d}_i) + \sum_{y\neq y_i}\log\sigma(-\mathbf{y}_y^T\mathbf{d}_i)]\\ & +\gamma\Omega(\mathbf{W},\mathbf{D},\mathbf{Y})\end{aligned}$$

5.2 Updating Rules

We use stochastic gradient descent method to train the proposed model. Thus for each training sample, we need to update the involved word, document or label representations. For a given training instance, $(w_i, w_j, d_k) \in \mathcal{P}$, Eq.(16) reduces to

$$f_1 = -\frac{1}{|\mathcal{P}|}\log\sigma(\mathbf{w}_j^T(\mathbf{w}_i+\mathbf{d}_k)) + \gamma(\|\mathbf{w}_i\|_2^2 + \|\mathbf{w}_j\|_2^2 + \|\mathbf{d}_k\|_2^2)$$

The derivatives of the above equation w.r.t. $\mathbf{w}_i, \mathbf{w}_j$ and $\mathbf{d}_k$ are given as

$$\begin{aligned}\nabla_{\mathbf{w}_i} f_1 &= \frac{1}{|\mathcal{P}|}[\sigma(\mathbf{w}_j^T(\mathbf{w}_i+\mathbf{d}_k)) - 1]\mathbf{w}_j + 2\gamma\mathbf{w}_i\\ \nabla_{\mathbf{d}_k} f_1 &= \frac{1}{|\mathcal{P}|}[\sigma(\mathbf{w}_j^T(\mathbf{w}_i+\mathbf{d}_k)) - 1]\mathbf{w}_j + 2\gamma\mathbf{d}_j\\ \nabla_{\mathbf{w}_j} f_1 &= \frac{1}{|\mathcal{P}|}[\sigma(\mathbf{w}_j^T(\mathbf{w}_i+\mathbf{d}_k)) - 1](\mathbf{w}_i+\mathbf{d}_k) + 2\gamma\mathbf{w}_j\end{aligned} \tag{17}$$

Then $\mathbf{w}_i, \mathbf{w}_j$ and $\mathbf{d}_k$ are updated as

$$\begin{aligned}\mathbf{w}_i &\leftarrow \mathbf{w}_i - \eta\nabla_{\mathbf{w}_i} f_1\\ \mathbf{d}_k &\leftarrow \mathbf{d}_k - \eta\nabla_{\mathbf{d}_k} f_1\\ \mathbf{w}_j &\leftarrow \mathbf{w}_j - \eta\nabla_{\mathbf{w}_j} f_1\end{aligned} \tag{18}$$

where η is the learning rate.

Similarly, when the training instance is $(w_i, w_t, d_k) \in \mathcal{N}$, Eq.(16) is reduced to:

$$f_2 = -\frac{1}{|\mathcal{N}|}\log\sigma(-\mathbf{w}_t^T(\mathbf{w}_i+\mathbf{d}_k)) + \gamma(\|\mathbf{w}_i\|_2^2 + \|\mathbf{w}_t\|_2^2 + \|\mathbf{d}_k\|_2^2)$$

Then $\mathbf{w}_i, \mathbf{w}_t$ and $\mathbf{d}_k$ are updated as

$$\begin{aligned}\mathbf{w}_i &\leftarrow \mathbf{w}_i - \eta\nabla_{\mathbf{w}_i} f_2\\ \mathbf{d}_k &\leftarrow \mathbf{d}_k - \eta\nabla_{\mathbf{d}_k} f_2\\ \mathbf{w}_t &\leftarrow \mathbf{w}_t - \eta\nabla_{\mathbf{w}_t} f_2\end{aligned} \tag{19}$$

When a training instance is from $\mathcal{E}$, say (d_i, d_j), we update $\mathbf{d}_i$ and $\mathbf{d}_j$ by the gradient descent method. When a training instance is from $\mathcal{N}_E$, say (d_i, d_t), we update $\mathbf{d}_i$ and $\mathbf{d}_t$. Similarly, when the training instance is (d_i, y_i), we update $\mathbf{d}_i$ and $\mathbf{Y}$ since all the label representation $\mathbf{Y}$ is involved as shown in the forth line of Eq.(16) . We omit the detailed derivations here since they are very similar to the aforementioned ones.

5.3 Subsampling of Frequent Words

There is another issue we need to deal with. In large corpora, the most frequent words such as ("in", "a") can easily occur millions of times. In each epoch, these words will be trained millions of times correspondingly and the vector representations of these words will not change significantly after several epochs [18]. On the contrary, some rare words are trained less frequently in each epoch thus they need more epochs to train. To account the imbalance between rare and frequent words, we use the sub-sampling approach as in [18]: each word w_i in the training set is discarded with a probability computed by the formula:

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \tag{20}$$

where $f(w_i)$ is the frequency of the word w_i and t is a chosen threshold, typically around 10^{-5}. The advantage of this subsampling formula is that it aggressively sub-samples words whose frequencies are greater than t while preserving the ranking of the frequencies.

5.4 A Learning Algorithm for LDE

With the negative sampling and the update rules, the algorithm to learn LDE is summarized in Algorithm 1. We first prepare the training instances from line 1 to line 7. In line 8, we initialize the parameters $\mathbf{W}, \mathbf{D}$ and $\mathbf{Y}$. Following the common practice, we initialize each element of $\mathbf{W}, \mathbf{D}$ and $\mathbf{Y}$ by randomly sampling from the uniform distribution [-0.2,0.2]. We then train LDE and update $\mathbf{W}, \mathbf{D}$ and $\mathbf{Y}$ given the training data using the gradient descent method from line 9 to line 13. Finally, the document embedding $\mathbf{D}$ and word embedding $\mathbf{W}$ are obtained.

$\mathbf{D}$ is the document embedding which we name LDE-Doc. We can also represent documents using word embeddings. In particular, to get the document representation from word-embeddings $\mathbf{W}$, for a document d_i, we average all the words in the document as

$$\tilde{\mathbf{d}}_i = \frac{1}{N_i}\sum_{w_i\in d_i}\mathbf{w}_i \tag{21}$$

where, N_i is the length of the document d_i and $\tilde{\mathbf{d}}_i$ is used as the document representation for d_i. We denote the document representation by word embedding as LDE-Word.

5.5 Time Complexity

When the training instance is $(w_i, w_j, d_k) \in \mathcal{P}$, from Eq.(17) and Eq.(18), we can see that the cost of calculating the derivative of f_1 w.r.t. $\mathbf{w}_i$ and updating $\mathbf{w}_i$ are both $\mathcal{O}(d)$.

Algorithm 1 LDE - Linked Document Embedding

```
Input: $\mathcal{D}, \mathcal{G} = \{\mathcal{V}, \mathcal{E}\}, \mathcal{Y}, \lambda$, window size $c$, dimension $d$
Output: $\mathbf{D}, \mathbf{W}$
 1: Construct $\mathcal{P}$ by using a sliding window size $c$ to extract instances as $(w_i, w_j, d_k)$ from documents where $(w_i, w_j, d_k)$ is added to $\mathcal{P}$ with the probability $\sqrt{\frac{t}{f(w_i)}}$
 2: for each training sample in $\mathcal{P}$ do
 3:     Draw $K$ negative samples from noise distribution and put to $\mathcal{N}$
 4: end for
 5: for $e_{ij}$ in $\mathcal{E}$ do
 6:     Randomly sample $K$ documents that are not linked with $d_i$ and put them into $\mathcal{N}_E$
 7: end for
 8: Initialize $\mathbf{W}, \mathbf{D}$ and $\mathbf{Y}$
 9: repeat
10:     for each training instance do
11:         Update involved parameters using SGD as described in Section 5.2
12:     end for
13: until Convergence
14: Return $\mathbf{D}$, $\mathbf{W}$
```

With similar analysis, we find that the computational cost of calculating gradients and updating parameters are also $\mathcal{O}(d)$ when training instances are from $\mathcal{N}, \mathcal{P}, \mathcal{E}, \mathcal{N}_E$ or $\mathcal{Y}$. Thus, we only need to count the size of the training data, which is $(K+1)|\mathcal{P}| + (K+1)|\mathcal{E}| + N_c|\mathcal{Y}|$. Therefore, the total computational cost in one epoch is $\big((K+1)|\mathcal{P}| + (K+1)|\mathcal{E}| + N_c|\mathcal{Y}|\big)\mathcal{O}(d)$. Considering the fact that $\mathcal{E}$ is usually very sparse, the complexity is comparable to Skip-gram, which is scalable to millions of documents [18].

6. EXPERIMENTS

In this section, we conduct experiments to evaluate the effectiveness of the proposed framework LDE. Specifically, we aim to answer the following questions:

- How effective is the proposed framework in learning document representations compared to the state-of-the-art methods?
- How does label information affect the performance of the proposed framework? and
- Does the network information provide additional information for learning better document representations?

We begin by introducing the datasets and experimental settings, and then we compare LDE with the state-of-the-art algorithms for classification to answer the fist question. We also investigate the sensitivity of LDE w.r.t. label and link information to answer the second and third questions.

6.1 Datasets and Experimental Settings

6.1.1 Datasets

The experiments are conducted on two real-world linked document datasets, DBLP and BlogCatalog. DBLP dataset is extracted by Arnetminer [24] from the DBLP website. Each document in DBLP dataset contains the title, authors and year of a paper. Some documents also contain venues, abstracts and reference papers. We use titles and abstracts as the document contents. Thus, we remove documents whose abstracts and titles are missing. We then choose six categories from the corpus[1], including "Computer networks", "Database:Data mining:Information retrieval", "Computer graphics:Multimedia", "Software engineering", "Theoretical computer science" and "High-Performance Computing". After that, we randomly select 2550 samples from each chosen category and add links between two documents if one document cites another document. BlogCatalog[2] is a blog directory where users can register their blogs under predefined categories. The categories are used as class labels of blogs. Each blog has a text description added by the owner, which is used as document content in our work. The homepage of each blog lists several blogs related to this blog, which forms links between a blog and its related blogs. In addition, if the owner of blog A follows the owner of blog B, we also add a link from blog A to blog B. We remove categories whose number of blogs are less than 500, which leaves us 27 categories and 62,652 blogs. Note that BlogCatalog dataset is unbalanced. For both datasets, we remove stop words and no further text normalizations such as stemming are done. The statistics of two datasets are summarized in Table 1.

Table 1: Statistics of the Datasets

Dataset	DBLP	BlogCatalog
# of documents	15,300	62,652
# of links	36,359	378,161
# of classes	6	27

6.1.2 Evaluation Metrics

Our goal is to learn vector presentations of documents for classification. Therefore, we use classification performance to assess the quality of learned document representations. In fact, the classification task is also a common way to evaluate these word and document embedding algorithms with unsupervised settings [13]. Two widely used classification evaluation metrics, Micro-F1 and Macro-F1, are adopted. The larger the Micro-F1 and Macro-F1 scores are, the better the document representation is for the classification task.

6.2 Performance Comparison

To answer the first question, we compare the proposed framework LDE with other classical and state-of-the-art document representation learning algorithms. Since LDA utilizes contents, links and labels during learning process, for fair comparison, the compared algorithms include state-of-the-art algorithms that utilizes links and contents such as RTM, TADW, contents and labels such as PTE, CNN and also graph-based classifier such as GC, which utilizes contents, link and labels for classification. The details of these algorithms are listed as follows:

- BOW [21]: the classical "bag-of-words" represent each document as a M-dimensional vector, where M is the size of the vocabulary and weight of each dimension is calculated by the TFIDF scheme.
- RTM [7]: relational topic model is an extension of topic modeling that models document content and links between documents.

[1]Categories are defined according to venue by Arnetminer
[2]https://www.blogcatalog.com/

Table 2: Document Classification Performance Comparison on DBLP and BlogCatalog

Dataset	DBLP		BlogCatalog	
Name	Micro-F1	Macro-F1	Micro-F1	Macro-F1
BOW	78.50±0.64	78.61±0.63	46.35±0.42	40.78±0.43
RTM	74.05±0.68	74.08±0.71	44.62±0.35	39.60±0.37
Skip-gram	81.00±0.40	80.98±0.41	47.38±0.28	41.97±0.25
CBOW	77.33±0.73	77.31±0.73	45.43±0.44	39.03±0.29
PV-DM	84.25±0.26	84.25±0.26	48.35±0.24	42.78±0.23
PV-DBOW	80.81±0.30	80.82±0.29	47.56±0.23	41.68±0.25
LP	72.88±0.75	72.90±0.76	38.54±0.42	35.51±0.40
GC	84.75±0.82	84.74±0.81	48.76±0.37	42.98±0.34
TADW	85.59±0.65	85.58±0.64	49.85±0.31	43.95±0.32
CNN	84.07±0.45	84.09±0.48	49.01±0.51	43.38±0.47
PTE	85.26±0.47	85.23±0.49	50.36±0.43	44.58±0.42
LDE-Word	80.87±0.36	80.83±0.39	48.77±0.29	42.96±0.25
LDE-Doc	**87.69±0.42**	**87.70±0.45**	**53.14±0.42**	**46.85±0.39**

- Skip-gram [18]: one of the state-of-the-art word embedding model and its training objective is to find word representations that are useful for predicting the surrounding words of a selected word in a sentence. After obtaining word embeddings by Skip-gram, we use Eq.(21) to get document representations.

- CBOW [18]: another state-of-the-art word embedding model. Unlike Skip-gram, the training objective of CBOW is to find word representations that are useful for predicting the center word by its neighbors. Similarly, we use Eq.(21) to get document representations.

- PV-DM [13]: the distributed memory version of paragraph vector which considers the order of the words. It aims at learning document embeddings that are good at predicting the next given context.

- PV-DBOW [13]: the distributed bag-of-words version of paragraph vector model proposed in [13]. Unlike PV-DM, the word order is ignored in PV-DBOW. It aims to learn document representations that are good at predicting words in the document.

- LP [34]: a traditional semi-supervised algorithm based on label propagation, which performs classification by propagating label information from labeled data to unlabeled data through the graph. LP denotes a traditional method that utilizes both network information and label information for classification.

- GC [4] a graph-based classification method which utilizes both document contents, link and label information into a probabilistic framework for classification.

- CNN [11]: convolution neural network for classification. It uses word embeddings as input to train convolution neural network with label information[3].

- TADW [32]: text-associated DeepWalk is a matrix factorization based method that utilizes both link and document data[4].

- PTE [23]: predictive text embedding which considers label information to learn word embedding but cannot handle link information among documents.

- **LDE**-Word: the proposed framework trains both word embedding and document embedding. This variant uses the average of the word embeddings to represent a document.

- **LDE**-Doc: the proposed framework. Instead of using the word embeddings, we use the document embeddings directly as the representations of the documents.

The experiment consists of two phases, i.e., the representation learning phase and the document classification phase. During the representation learning phase, all the documents in the training set and testing set are used to learn word embeddings or document embeddings. For LDE-Word and LDE-Doc, labels of training data and link information are also used for learning embeddings during the representation learning phase. Labels of testing data are held out and no algorithm can use labels of testing data during the representation learning phase. During the classification phase, we use libsvm[5] [6] to train a SVM classifier using the learned document embeddings and the training data. The trained SVM classifier is then assessed on the testing data[6].

There are some parameters to set for the baseline algorithms. For a fair comparison, for Skip-gram, CBOW, PV-DM, PV-DBOW, CNN, RTM and LDE, we set the embedding dimension to be 100. For Skip-gram, CBOW, PV-DM, PV-DBOW and LDE, following the parameter setting suggestions in [18], we set the window size to be 7 and the number of negative samples also to be 7. We follow the setting in [23] for PTE and we use the default setting in the code of TADW. For the proposed model, we choose γ to be 0.0001. As of CNN, we use the default architecture in [11]. For both datasets, we randomly select 60% as training data and the remaining 40% as testing data. The random selection is conducted 5 times and the average micro-f1 and macro-f1 with standard deviations are reported in Table 2. From the table, we make the following observations:

- Skip-gram and PV-DM outperforms BOW slightly on both datasets. This shows that word/document embeddings can learn dense representations of documents which can improve the classification performance slightly.

[3]Code available at https://github.com/yoonkim/CNN_sentence
[4]We use the code from https://github.com/albertyang33/TADW

[5]Avaliable at https://www.csie.ntu.edu.tw/ cjlin/libsvm/
[6]For Skip-gram, CBOW, PV-DM and PV-DBOW, we use the implementation by Gensim, which is available at https://radimrehurek.com/gensim/

Table 3: Effects of Label Density for LDE.

Dataset	Algorithm	Metrics	0%	20%	40%	60%	80%	100%
DBLP	LDE-Word	Micro-F1	76.62	77.30	78.25	78.79	79.69	80.87
		Macro-F1	76.63	77.26	78.23	78.76	79.65	80.83
	LDE-Doc	Micro-F1	78.68	79.05	81.85	83.87	85.65	87.69
		Macro-F1	78.68	79.04	81.85	83.87	85.65	87.70
BlogCatalog	LDE-Word	Micro-F1	45.45	45.92	46.36	47.05	47.98	48.77
		Macro-F1	39.62	40.09	40.73	41.47	42.20	42.96
	LDE-Doc	Micro-F1	46.57	47.12	48.83	50.17	51.62	53.14
		Macro-F1	40.65	41.24	42.76	44.15	45.51	46.85

In contrast, LDE-Doc is much better than BOW on both datasets, which demonstrates the effectiveness of the proposed framework by incorporating link and label information.

- Most of the time, CNN outperforms other baseline methods. CNN uses label information for training and it is likely to obtain better performance. PTE outperforms CNN, which is consistent with previous observations [23].

- Comparing LDE-Doc and TADW, we can see that the performance of LDE-Doc is better than TADW. This is because though TADW utilizes link information, it doesn't consider label information for learning document representation.

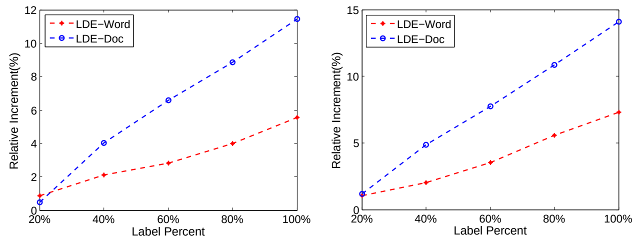
- The performance of LDE-Doc is much better than LDE-Word. This is because LDE focuses on learning document representations. The link information and label information are used by LDE specific to document embedding instead of word embedding. LDE-Word is comparable to Skip-gram.

- The performance of LDE-Doc is better than the graph-based classification method GC, which also utilizes contents, link and label information for classification. This suggests that by utilizing the distributional hypothesis idea and exploiting the word-word-doc, doc-link and doc-label relationships, the learned document representations is good for classification.

- Though both PTE and LDE-Doc follow the idea of distributional hypothesis and use the label information, LDE-Doc significantly outperforms PTE. This is because in addition to label information, LDE-Doc also models the link information among documents which is pervasively available for linked documents.

- The proposed document embedding algorithm LDE-Doc outperforms representative document representation algorithms including PV-DM, RTM, PV-DBOW, PTE and TADW and graph-classification based methods such as GC, which further demonstrates that by considering link and label information, the proposed framework LDE is able to learn better document representations.

We conduct t-test on all performance comparisons and it is evident from t-test that all improvements are significant. In summary, the proposed framework can learn better document embeddings for classification by exploiting link and label information.



(a) Micro-F1 on DBLP (b) Micro-F1 on BlogCatalog

Figure 2: Relative performance improvement of LDE with increasing label information.

6.3 Impact of Label Density

In this subsection, we perform experiments to investigate the effects of the label density on the quality of word and document embeddings. We first split each dataset into 60% and 40%, where 40% is fixed as testing data. From the 60% part, we sample $x\%$ as labeled data in the embedding learning phase. The remaining (100-x)% are not used for learning the embeddings. We vary x as $\{0, 20, 40, 60, 80, 100\}$. Similarly, the experiment is composed of two phases. During the representation learning phase, we use all the documents with the $x\%$ labeled data and link information to learn the embeddings. After the embeddings are learned, we use libsvm to train an SVM classifier. Each experiment is conducted 5 times and we report the average classification performance in terms of Micro-F1 and Macro-F1 in Table 3. To help understand the effects, we plot the relative performance improvement compared to that without label information ($x = 0$) in terms of Micro-F1 in Figure 2. Note that we omit the figure for Macro-F1 since we have similar observations. From the table and the figures, we make the following observations:

- For both datasets, with the increase of labeled data in the representation learning phase, the performance of both word embedding and document embedding increases, which demonstrates that by incorporating label information, we can learn better document and word embeddings.

- From the figure, we can see that with the increase of labeled data, the difference between LDE-Word and LDE-Doc also increases, which indicates that label information is more useful for the proposed framework to learn document embeddings than word embeddings. This is reasonable because we explicitly model document and label relations to enable the capability of the learned representations in predicting labels.

6.4 Effects of Link Density

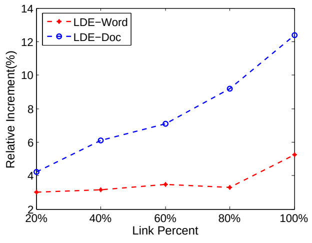
In this subsection, we perform experiment to investigate the effects of the link density on the quality of word and doc-

Table 4: Effects of Link Density on LDE.

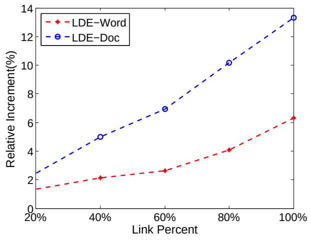
Dataset	Algorithm	Metrics	0%	20%	40%	60%	80%	100%
DBLP	LDE-Word	Micro-F1	76.83	79.15	79.25	79.51	79.35	80.87
		Macro-F1	76.82	79.12	79.23	79.48	79.32	80.83
	LDE-Doc	Micro-F1	78.03	81.31	82.78	83.56	85.20	87.69
		Macro-F1	78.05	81.30	82.79	83.57	85.22	87.70
BlogCatalog	LDE-Word	Micro-F1	45.87	46.48	46.84	47.07	47.75	48.77
		Macro-F1	39.87	40.38	40.97	41.52	42.13	42.96
	LDE-Doc	Micro-F1	46.89	48.04	49.23	50.15	51.65	53.14
		Macro-F1	40.91	41.62	42.65	44.27	45.71	46.85

ument embeddings. Each time, we randomly sample $x\%$ of links. We vary x as $\{0, 20, 40, 60, 80, 100\}$. We then split the dataset into 60% and 40%, where 60% are used for training and 40% are used for testing. The experiment is composed of two phases. During the representation learning phase, we use all the documents, the label for the training data and the $x\%$ of links to learn embeddings. After learning embeddings, we use libsvm to train a SVM classifier. We report the classification performance in terms of Micro-F1 and Macro-F1 in Table 4. Similarly, to help understand the effects, we plot the relative performance improvement in terms of micro-f1 w.r.t. compared to that without links ($x = 0$) in Figure 3. From the table and the figures, we make the following observations:

- For both datasets, as the percentage of links increases during the representation learning phase, the performance of both word embedding and document embedding increases, which demonstrates that by incorporating link information, we can learn better document and word embeddings.
- From the figure, we can see that as the percentage of links increases, the difference between LDE-Word and LDE-Doc also increases, which suggests that link information helps document embedding more than word embedding. The reason is that we extract document and document relations from link information and then explicitly model them based on document embeddings.



(a) Micro-F1 on DBLP

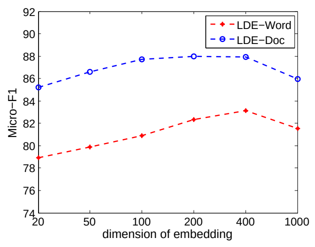


(b) Micro-F1 on BlogCatalog

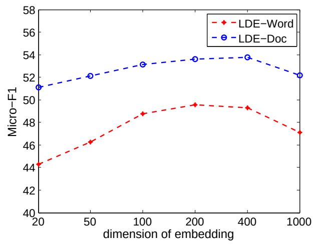
Figure 3: Relative performance improvement of LDE with increasing link information.

6.5 Effects of Embedding Dimensions

In this subsection, we investigate how the embedding dimensions affect the performance of the proposed framework LDE. In detail, we first randomly select 60% as training and the remaining 40% as testing. All the documents, link information and label information of the 60% training data are used for learning document and word embeddings. After that, we train a SVM classifier to perform document classification with the learned document and word embeddings on the testing data. We vary the number of embedding dimension d as $\{20, 50, 100, 200, 400, 1000\}$. The random selection process is done 5 times and the average Micro-F1 are shown in Figure 4. Note that we only report Micro-F1 since the performance in terms of Macro-F1 is very close to Micro-F1. From the figures, we note that as the dimension of embeddings increases, the performance of both document embedding and word embedding first increases and then decreases. This is because when the embedding dimension is too small, the representation capability of the embedding vectors is not sufficient and we may lose information. However, when the embedding dimension is too large, the model is too complex and we may overfit to the data.



(a) Micro-F1 on DBLP



(b) Micro-F1 on BlogCatalog

Figure 4: The effects of embedding dimension on the classification performance of document and word embedding.

7. CONCLUSION

In many real-word applications, documents are linked via various means. Little work exists for exploiting link information for document embedding. In this paper, we investigate the problem of linked document embedding for classification. We propose a novel framework LDE that captures content, link and label information into a coherent model for learning document and word embeddings simultaneously. Experimental results on real-world datasets show that the proposed framework outperforms state-of-the-art document representation algorithms for classification. Further experiments are conducted to demonstrate the effects of label density and link density on the performance of LDE, which suggest that both link and label information can help learn better word and document embeddings for classification.

There are several interesting directions that need further investigation. First, in this work, we consider linked document embedding for classification and it will be interesting to investigate linked document embedding specific to other tasks such as ranking or recommendation [30]. Second, currently we deal with unsigned and unweighted links and we would like to investigate how to extend the proposed framework to handle other types of links such as signed and weighted links [25].

8. ACKNOWLEDGMENTS

This work is sponsored, in part, by Office of Naval Research (ONR) grant N00014-16-1-2257 and National Science Foundation (NSF) under grant number IIS-1217466.

9. REFERENCES

[1] Jacob Abernethy, Olivier Chapelle, and Carlos Castillo. Graph regularization methods for web spam detection. *Machine Learning*, 81(2):207–225, 2010.

[2] Ralitsa Angelova, Gjergji Kasneci, and Gerhard Weikum. Graffiti: graph-based classification in heterogeneous networks. *World Wide Web*, 15(2):139–170, 2012.

[3] Ralitsa Angelova and Stefan Siersdorfer. A neighborhood-based approach for clustering of linked document collections. In *CIKM*. ACM, 2006.

[4] Ralitsa Angelova and Gerhard Weikum. Graph-based text classification: learn from your neighbors. In *SIGIR*, pages 485–492. ACM, 2006.

[5] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[6] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM TIST*, 2(3):27, 2011.

[7] Jonathan Chang and David M Blei. Relational topic models for document networks. In *AISTAS*, 2009.

[8] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. Heterogeneous network embedding via deep architectures. In *Proceedings of KDD*. ACM, 2015.

[9] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[10] John R Firth. A synopsis of linguistic theory 1930–55 (special volume of the philological society), 1957.

[11] Yoon Kim. Convolutional neural networks for sentence classification. *EMNLP*, 2014.

[12] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[13] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196, 2014.

[14] Bo Long, Zhongfei Mark Zhang, and Philip S Yu. A probabilistic framework for relational clustering. In *KDD*. ACM, 2007.

[15] Hao Ma, Michael R Lyu, and Irwin King. Learning to recommend with trust and distrust relationships. In *RecSys*, pages 189–196. ACM, 2009.

[16] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *RecSys*. ACM, 2007.

[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[18] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[19] Jennifer Neville, Micah Adler, and David Jensen. Clustering relational data using attribute and link information. In *Proceedings of the text mining and link analysis workshop, 18th international joint conference on artificial intelligence*, pages 9–15, 2003.

[20] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. Deep sentence embedding using the long short term memory network: Analysis and application to information retrieval. *arXiv preprint arXiv:1502.06922*, 2015.

[21] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[22] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

[23] Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *KDD*. ACM, 2015.

[24] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *KDD*, pages 990–998. ACM, 2008.

[25] Jiliang Tang, Yi Chang, Aggarwal Charu, and Huan Liu. A survey of signed network mining in social media. *ACM Computing Survey*, 2016.

[26] Jiliang Tang and Huan Liu. Feature selection with linked data in social media. In *SDM*. SIAM, 2012.

[27] Jiliang Tang and Huan Liu. Unsupervised feature selection for linked social media data. In *KDD*, 2012.

[28] Mihaela Vela and Liling Tan. Predicting machine translation adequacy with document embeddings. *EMNLP*, page 402, 2015.

[29] Suhang Wang, Jiliang Tang, Fred Morstatter, and Huan Liu. Paired restricted boltzmann machine for linked data. In *CIKM*. ACM, 2016.

[30] Suhang Wang, Jiliang Tang, Yilin Wang, and Huan Liu. Exploring implicit hierarchical structures for recommender systems. In *IJCAI*, 2015.

[31] Xing Wei and W Bruce Croft. Lda-based document models for ad-hoc retrieval. In *SIGIR*. ACM, 2006.

[32] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. Network representation learning with rich text information. In *IJCAI*, 2015.

[33] Shenghuo Zhu, Kai Yu, Yun Chi, and Yihong Gong. Combining content and link for classification using matrix factorization. In *SIGIR*. ACM, 2007.

[34] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Citeseer, 2002.

[35] Yaojia Zhu, Xiaoran Yan, Lise Getoor, and Cristopher Moore. Scalable text and link analysis with mixed-topic link models. In *KDD*, 2013.