PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks

Jian Tang
Microsoft Research Asia
jiatang@microsoft.com

Meng Qu*
Peking University
mnqu@pku.edu.cn

Qiaozhu Mei
University of Michigan
qmei@umich.edu

ABSTRACT

Unsupervised text embedding methods, such as Skip-gram and Paragraph Vector, have been attracting increasing attention due to their simplicity, scalability, and effectiveness. However, comparing to sophisticated deep learning architectures such as convolutional neural networks, these methods usually yield inferior results when applied to particular machine learning tasks. One possible reason is that these text embedding methods learn the representation of text in a fully unsupervised way, without leveraging the labeled information available for the task. Although the low dimensional representations learned are applicable to many different tasks, they are not particularly tuned for any task. In this paper, we fill this gap by proposing a semi-supervised representation learning method for text data, which we call the *predictive text embedding* (PTE). Predictive text embedding utilizes both labeled and unlabeled data to learn the embedding of text. The labeled information and different levels of word co-occurrence information are first represented as a large-scale heterogeneous text network, which is then embedded into a low dimensional space through a principled and efficient algorithm. This low dimensional embedding not only preserves the semantic closeness of words and documents, but also has a strong predictive power for the particular task. Compared to recent supervised approaches based on convolutional neural networks, predictive text embedding is comparable or more effective, much more efficient, and has fewer parameters to tune.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation

*This work was done when the second author was an intern at Microsoft Research Asia.

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.
© 2015 ACM. ISBN 978-1-4503-3664-2/15/08 ...$15.00.
DOI: http://dx.doi.org/10.1145/2783258.2783307.

Keywords

predictive text embedding, representation learning

1. INTRODUCTION

Learning a meaningful and effective representation of text, e.g., for words and documents, is a critical prerequisite for many machine learning tasks such as text classification, clustering and retrieval. Traditionally, every word is represented independently to each other, and each document is represented as a "bag-of-words". However, both representations suffer from problems such as data sparsity, polysemy, and synonymy, as the semantic relatedness between different words are commonly ignored.

Distributed representations of words and documents [18, 10] effectively address this problem through representing words and documents in low-dimensional spaces, in which similar words and documents are embedded closely to each other. The essential idea of these approaches comes from the distributional hypothesis that "you shall know a word by the company it keeps" (Firth, J.R. 1957:11) [7]. Mikilov et al. proposed a simple and elegant word embedding model called the Skip-gram [18], which uses the embedding of the target word to predict the embedding of each individual context word in a local window. Le and Mikolov further extended this idea and proposed the Paragraph Vectors [10] in order to embed arbitrary pieces of text, e.g., sentences and documents. The basic idea is to use the embeddings of sentences/documents to predict the embeddings of words in the sentences/documents. Comparing to other classical approaches that also utilize the distributional similarity of word context, such as the Brown clustering or nearest neighbors, these text embedding approaches have been proved to be quite efficient, scaling up to millions of documents on a single machine [18].

Because of the unsupervised learning process, the representations learned through these text embedding models are general enough and can be applied to a variety of tasks such as classification, clustering and ranking. However, when compared end-to-end with sophisticated deep learning approaches such as the convolutional neural networks (CNNs) [5, 8], the performance of text embeddings usually falls short on specific tasks [30]. This is perhaps not surprising as the deep neural networks fully leverage labeled information that is available for a task when they learn the representations of the data. Most text embedding methods are not able to consider labeled information when learning the representations; the labels only kick in later when a classifier is trained using the representations as features. In other words, unsuper-

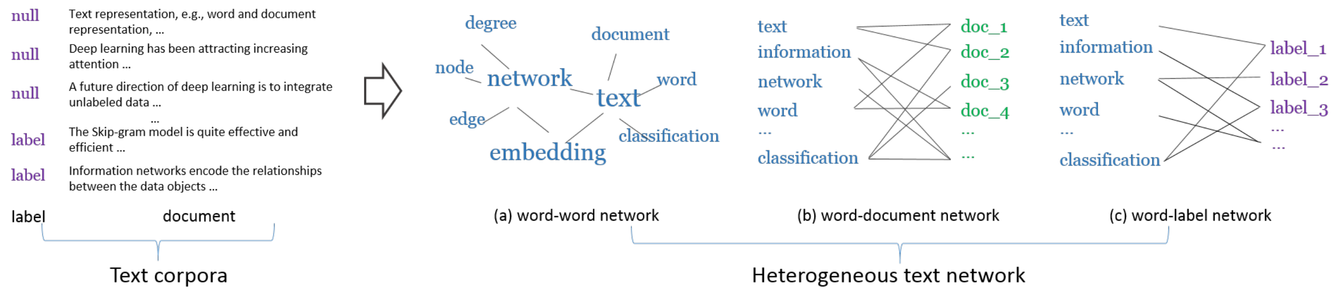

null
Text representation, e.g., word and document representation, ...
null
Deep learning has been attracting increasing attention ...
null
A future direction of deep learning is to integrate unlabeled data ...
label
The Skip-gram model is quite effective and efficient ...
label
Information networks encode the relationships between the data objects ...
label
document
Text corpora
degree
document
node
network
word
text
edge
classification
embedding
(a) word-word network
text
information
network
word
classification
doc_1
doc_2
doc_3
doc_4
(b) word-document network
text
information
network
word
classification
label_1
label_2
label_3
(c) word-label network
Heterogeneous text network

Figure 1: Illustration of converting a partially labeled text corpora to a heterogeneous text network. The word-word co-occurrence network and word-document network encode the unsupervised information, capturing the local context-level and document-level word co-occurrences respectively; the word-label network encodes the supervised information, capturing the class-level word co-occurrences.

vised text embeddings are generalizable for different tasks but have a weaker predictive power for a particular task.

Despite this deficiency, there are still considerable advantages of text embedding approaches comparing to deep neural networks. First, the training of deep neural networks, especially convolutional neural networks is computational intensive, which usually requires multiple GPUs or clusters of CPUs when processing a large amount of data; second, convolutional neural networks usually assume the availability of a large amount of labeled examples which is unrealistic in many tasks. The easily obtainable unlabeled data are usually used through an indirect way of pre-training; third, the training of CNNs requires exhaustive tuning of many parameters, which is very time consuming even for experts and infeasible for non-experts. On the other hand, text embedding methods like Skip-gram are much more efficient, are much easier to tune, and naturally accommodate unlabeled data.

In this paper, we fill this gap by proposing the predictive text embedding (PTE), which adapts the advantages of unsupervised text embeddings but naturally utilizes labeled information in representation learning. With predictive text embedding, an effective low dimensional representation is learned jointly from limited labeled examples and a large amount of unlabeled examples. Comparing to unsupervised embeddings, this representation is optimized for particular tasks like what convolutional neural networks do (i.e., the representation has strong predictive power for the particular classification task).

The proposed method naturally extends our previous work of unsupervised information network embedding [27] and first learns a low dimensional embedding for words through a heterogeneous text network. The network encodes different levels of co-occurrence information between words and words, words and documents, and words and labels. The network is embedded into a low dimensional vector space that preserves the second-order proximity [27] between the vertices in the network. The representation of an arbitrary piece of text (e.g., a sentence or a document) can be simply inferred as the average of the word representations, which turns out to be quite effective. The whole optimization process remains very efficient, which scales up to millions of documents and billions of tokens on a single machine.

We conduct extensive experiments with real-world text corpora, including both long and short documents. Experimental results show that the predictive text embeddings significantly outperform the state-of-the-art unsupervised embeddings in various text classification tasks. Compared end-to-end with convolutional neural networks for text classification [8], predictive text embedding outperforms on long documents and generates comparable results on short documents. PTE enjoys various advantages over convolutional neural networks as it is much more efficient, accommodates large-scale unlabeled data effectively, and is less sensitive to model parameters. We believe our exploration points to a direction of learning text embeddings that could compete head-to-head with deep neural networks in particular tasks.

To summarize, we make the following contributions:

- We propose to learn predictive text embeddings in a semi-supervised manner. Unlabeled data and labeled information are integrated into a heterogeneous text network which incorporates different levels of co-occurrence information in text.

- We propose an efficient algorithm "PTE", which learns a distributed representation of text through embedding the heterogeneous text network into a low dimensional space. The algorithm is very efficient and has few parameters to tune.

- We conduct extensive experiments using various real-world data sets and compare predictive text embedding end-to-end with both unsupervised text embeddings and convolutional neural networks.

The rest of this paper is organized as follows. We first introduce the related work in Section 2. Section 3 formally defines the problem of predictive text embedding through heterogeneous text networks. Section 4 introduces the proposed algorithm in details. Section 5 presents the results of empirical experiments. We conclude in Section 6.

2. RELATED WORK

Our work is mainly related to distributed text representation learning and information network embedding.

2.1 Distributed Text Embedding

Distributed representation of text has proved to be quite effective in many natural language processing tasks such as word analogy [18], POS tagging [6], parsing [6], language modeling [17], and sentiment analysis [16, 10, 5, 8]. Existing approaches can be generally classified into two categories: unsupervised and supervised. Recent developed unsupervised approaches normally learn the embeddings of words and/or documents by utilizing word co-occurrences in the local context (e.g., Skip-gram [18]) or at document level (e.g., paragraph vectors [10]). These approaches are quite efficient, scaling up to millions of documents. The supervised approaches [5, 8, 23, 6] are usually based on deep neural network architectures, such as recursive neural tensor networks (RNTNs) [24] or convolutional neural networks (CNNs) [11]. In RNTNs, each word is embedded into a low dimensional vector, and the embeddings of the phrases are recursively learned by applying the same tensor-based composition function over the sub-phrases or words in a parse tree. In CNNs [5], each word is also represented with a vector, and the same convolutional kernel is applied over the context windows in different positions of the sentences, followed by a max-pooling and fully connected layer.

The major difference between these two categories of approaches is how they utilize labeled and unlabeled information in the representation learning phase. The unsupervised methods do not include labeled information when learning the representations and only use the labels to train the classifier after the data is transformed into the learned representation. RNTNs and CNNs incorporate the labels directly into representation learning, so the learned representations are particularly tuned for the classification task. To incorporate unlabeled examples, however, these neural nets usually have to use an indirect approach such as to pretrain the word embeddings with unsupervised approaches. Comparing to these two lines of work, PTE learns the text vectors in a semi-supervised way - the representation learning algorithm directly utilizes both labeled information and large-scale unlabeled data.

Another piece of work similar to predictive word embedding is [15], which learns word vectors that are particularly tuned for sentiment analysis. However, their approach does not scale to millions of documents and does not generalize to other classification tasks.

2.2 Information Network Embedding

Our work is also related to the problem of network/graph embedding as the word representations of PTE are learned through a heterogeneous text network. Embedding networks/graphs into low dimensional spaces is very useful in a variety of applications, e.g., node classification [3] and link prediction [13]. Classical graph embedding algorithms such as MDS [9], IsoMap [28] and Laplacian eigenmap [1] are not applicable for embedding large-scale networks that contain millions of vertices and billions of edges. There are some recent work attempting to embed very large real-world networks. Perozzi et al. [20] proposed a network embedding model called the "DeepWalk," which uses truncated random walks on the networks and is only applicable for networks with binary edges. Our previous work proposed a novel large-scale network embedding model called the "LINE," which is suitable for arbitrary types of information networks: undirected or directed, binary or weighted [27]. The LINE model optimizes an objective function which aims to preserve both the local and global network structures. Both Deepwalk and LINE are unsupervised and only handle homogeneous networks. The network embedding algorithm used by PTE extends the LINE to deal with heterogeneous networks, in which multiple types of vertices (including the class labels) and edges exist.

3. PROBLEM DEFINITION

Let us begin with formally defining the problem of predictive text embedding through heterogeneous text networks. Comparing to unsupervised text embedding approaches including Skip-gram and Paragraph Vectors that learn general semantic representations of text, our goal is to learn a representation of text that is optimized for a given text classification task. In other words, we anticipate the text embedding to have a strong predictive power of the performance of the given task. The basic idea is to incorporate both the labeled and unlabeled information when learning the text embeddings. To achieve this, it is desirable to first have an unified representation to encode both types of information. In this paper, we propose different types of networks to achieve this, including word-word co-occurrence networks, word-document networks, and word-label networks.

DEFINITION 1. ***(Word-Word Network)*** ***Word-word*** *co-occurrence network, denoted as $G_{ww} = (\mathcal{V}, E_{ww})$, captures the word co-occurrence information in local contexts of the unlabeled data. $\mathcal{V}$ is a vocabulary of words and E_{ww} is the set of edges between words. The weight w_{ij} of the edge between word v_i and v_j is defined as the number of times that the two words co-occur in the context windows of a given window size.*

The word-word network captures the word co-occurrences in local contexts, which is the essential information used by existing word embedding approaches such as Skip-gram. Beyond the local contexts, word co-occurrence at the document level is also widely explored in classical text representations such as statistical topic models, e.g., the latent Dirichlet allocation [4]. To capture the document-level word co-occurrences, we introduce another network, word-document network, defined as below:

DEFINITION 2. ***(Word-Document Network)*** ***Word-document*** *network, denoted as $G_{wd} = (\mathcal{V} \cup \mathcal{D}, E_{wd})$, is a bipartite network where $\mathcal{D}$ is a set of documents and $\mathcal{V}$ is a set of words. E_{wd} is the set of edges between words and documents. The weight w_{ij} between word v_i and document d_j is simply defined as the number of times v_i appears in document d_j.*

The word-word and word-document networks encode the unlabeled information in large-scale corpora, capturing word co-occurrences at both the local context level and the document level. To encode the labeled information, we introduce the word-label network, which captures word co-occurrences at category-level .

DEFINITION 3. ***(Word-Label Network)*** ***Word-label*** *network, denoted as $G_{wl} = (\mathcal{V} \cup \mathcal{L}, E_{wl})$, is a bipartite network that captures category-level word co-occurrences. $\mathcal{L}$ is a set of class labels and $\mathcal{V}$ a set of words. E_{wl} is a set of edges between words and classes. The weight w_{ij} of the edge between word v_i and class c_j is defined as: $w_{ij} = \sum_{(d:l_d=j)} n_{di}$, where n_{di} is the term frequency of word v_i in document d, and l_d is the class label of document d.*

The three types of networks above can be further integrated into one heterogeneous text network.

DEFINITION 4. ***(Heterogeneous Text Network)*** *The* ***heterogeneous text network*** *is the combination of word-word,*

word-document, and word-label networks constructed from both unlabeled and labeled text data. It captures different levels of word co-occurrences and contains both labeled and unlabeled information.

Note that the definition of a heterogeneous text network can be generalized to integrate other types of networks such as word-sentence, word-paragraph, and document-label networks. In this work we are using the three types of networks (word-word, word-document, and word-label) as an illustrative example. We particularly focus on word networks in order to first represent words into low dimensional spaces. The representation of other text units (e.g., sentences or paragraphs) can be then computed through aggregating the word representations.

Finally, we formally define the problem of predictive text embedding as follows:

DEFINITION 5. ***(Predictive Text Embedding)*** *Given a large collection of text data with unlabeled and labeled information, the problem of* ***predictive text embedding*** *aims to learn low dimensional representations of words by embedding the heterogeneous text network constructed from the collection into a low dimensional vector space.*

4. PREDICTIVE TEXT EMBEDDING

In this section, we introduce the proposed method that learns predictive text embedding through heterogeneous text networks. Our method first learns vector representations of words by embedding the heterogeneous text networks constructed from free text into a low dimensional space, and then infer text embeddings based on the learned word vectors. As the heterogeneous text network is composed of three bipartite networks, we first introduce an approach for embedding individual bipartite networks.

4.1 Bipartite Network Embedding

In our previous work, we introduced the LINE model to learn the embedding of large-scale information networks [27]. LINE is mainly designed for homogeneous networks, i.e., networks with the same types of nodes. LINE cannot be directly applied to heterogeneous networks as the weights on different types of edges are not comparable. Here, we first adapt the LINE model for embedding bipartite networks. The essential idea is to make use of the second-order proximity [27] between vertices, which assumes vertices with similar neighbors are similar to each other and thus should be represented closely in a low dimensional space.

Given a bipartite network $G = (\mathcal{V}_A \cup \mathcal{V}_B, E)$, where $\mathcal{V}_A$ and $\mathcal{V}_B$ are two disjoint sets of vertices of different types, and E is the set of edges between them. We first define the conditional probability of vertex v_i in set $\mathcal{V}_A$ generated by vertex v_j in set $\mathcal{V}_B$ as:

$$p(v_i|v_j) = \frac{\exp(\vec{u}_i^T \cdot \vec{u}_j)}{\sum_{i' \in A} \exp(\vec{u}_{i'}^T \cdot \vec{u}_j)}, \tag{1}$$

where $\vec{u}_i$ is the embedding vector of vertex v_i in $\mathcal{V}_A$, and $\vec{u}_j$ is the embedding vector of vertex v_j in $\mathcal{V}_B$. For each vertex v_j in $\mathcal{V}_B$, Eq (1) defines a conditional distribution $p(\cdot|v_j)$ over all the vertices in the set $\mathcal{V}_A$; for each pair of vertices $v_j, v_{j'}$, the second-order proximity can actually be determined by their conditional distributions $p(\cdot|v_j), p(\cdot|v_{j'})$. To preserve the second-order proximity, we can make the conditional distribution $p(\cdot|v_j)$ be close to its empirical distribution $\hat{p}(\cdot|v_j)$, which can be achieved by minimizing the following objective function:

$$O = \sum_{j \in B} \lambda_j d(\hat{p}(\cdot|v_j), p(\cdot|v_j)), \tag{2}$$

where $d(\cdot, \cdot)$ is the KL-divergence between two distributions, λ_j is the importance of vertex v_j in the network, which can be set as the degree $deg_j = \sum_i w_{ij}$, and the empirical distribution can be defined as $\hat{p}(v_i|v_j) = \frac{w_{ij}}{deg_j}$. Omitting some constants, the objective function (2) can be calculated as:

$$O = -\sum_{(i,j) \in E} w_{ij} \log p(v_j|v_i). \tag{3}$$

The objective (3) can be optimized with stochastic gradient descent using the techniques of edge sampling [27] and negative sampling [18]. In each step, a binary edge $e = (i, j)$ is sampled with the probability proportional to its weight w_{ij}, and meanwhile multiple negative edges (i, j) are sampled from a noise distribution $p_n(j)$. The sampling procedures address significant deficiency of stochastic gradient descent in learning network embeddings. For the detailed optimization process, readers can refer to [27].

The embeddings of the word-word, word-document, and word-label network can all be learned by the above model. Note that the word-word network is essentially a bipartite-network by treating each undirected edge as two directed edges, and then $\mathcal{V}_A$ is defined as the set of the source nodes, $\mathcal{V}_B$ as the set of target nodes. Therefore, we can define the conditional probabilities $p(v_i|v_j)$, $p(v_i|d_j)$ and $p(v_i|l_j)$ according to equation (1), and then learn the embeddings by optimizing objective function (3). Next, we introduce our approach of embedding the heterogeneous text network.

4.2 Heterogeneous Text Network Embedding

The heterogeneous text network is composed of three bipartite networks: word-word, word-document and word-label networks, where the word vertices are shared across the three networks. To learn the embeddings of the heterogeneous text network, an intuitive approach is to collectively embed the three bipartite networks, which can be achieved by minimizing the following objective function:

$$O_{pte} = O_{ww} + O_{wd} + O_{wl}, \tag{4}$$

where

$$O_{ww} = -\sum_{(i,j) \in E_{ww}} w_{ij} \log p(v_i|v_j) \tag{5}$$

$$O_{wd} = -\sum_{(i,j) \in E_{wd}} w_{ij} \log p(v_i|d_j) \tag{6}$$

$$O_{wl} = -\sum_{(i,j) \in E_{wl}} w_{ij} \log p(v_i|l_j) \tag{7}$$

The objective function (4) can be optimized in different ways, depending on how the labeled information, i.e., the word-label network, is used. One solution is to train the model with the unlabeled data (the word-word and word-document networks) and the labeled data simultaneously.

We call this approach *joint training.* An alternative solution is to learn the embeddings with unlabeled data first, and then fine-tune the embeddings with the word-label network. This is inspired by the idea of *pre-training and fine-tuning* in the literature of deep learning [2].

In joint training, all three types of networks are used together. A straightforward solution to optimize the objective (4) is to merge the all the edges in the three sets E_{ww}, E_{wd}, E_{wl} and then deploy edge sampling [27], which samples an edge for model updating in each step, with the sampling probability proportional to its weight. However, when the network is heterogeneous, the weights of the edges between different types of vertices are not comparable to each other. A more reasonable solution is to alternatively sample from the three sets of edges. We summarize the detailed training algorithm in Alg. 1.

Algorithm 1: Joint training.

```
Data: $G_{ww}, G_{wd}, G_{wl}$, number of samples $T$, number of
      negative samples $K$.
Result: word embeddings $\vec{w}$.
while $iter \leq T$ do
    • sample an edge from $E_{ww}$ and draw $K$ negative edges,
      and update the word embeddings;
    • sample an edge from $E_{wd}$ and draw $K$ negative edges,
      and update the word and document embeddings;
    • sample an edge from $E_{wl}$ and draw $K$ negative edges,
      and update the word and label embeddings;
end
```

Algorithm 2: Pre-training + Fine-tuning.

```
Data: $G_{ww}, G_{wd}, G_{wl}$, number of samples $T$, number of
      negative samples $K$.
Result: word embeddings $\vec{w}$.
while $iter \leq T$ do
    • sample an edge from $E_{ww}$ and draw $K$ negative edges,
      and update the word embeddings;
    • sample an edge from $E_{wd}$ and draw $K$ negative edges,
      and update the word and document embeddings;
end
while $iter \leq T$ do
    • sample an edge from $E_{wl}$ and draw $K$ negative edges,
      and update the word and label embeddings;
end
```

Similarly, we summarize the training process of pre-training and fine-tuning in Alg. 2.

4.3 Text Embedding

The heterogeneous text network encodes word co-occurrences at different levels, extracted from both unlabeled data and labeled information for a specific classification task. Therefore, the word representations learned by embedding the heterogeneous text network are not only more robust but also optimized for that task. Once the word vectors are learned, the representation of an arbitrary piece of text can be obtained by simply averaging the vectors of the words in that piece of text. That is, the vector representation of a piece of text $d = w_1 w_2 \cdots, w_n$ can be computed as

$$\vec{d} = \frac{1}{n}\sum_{i=1}^{n} \vec{u}_i, \tag{8}$$

where $\vec{u}_i$ is the embedding of word w_i.

In fact, the average of the word embeddings is the solution to minimizing the following objective function:

$$O = \sum_{i=1}^{n} l(\vec{u}_i, \vec{d}), \tag{9}$$

where the loss function $l(\cdot,\cdot)$ between the word embedding $\vec{u}_i$ and text embedding $\vec{d}$ is specified as the Euclidean distance. Related is the inference process of paragraph vectors [10], which minimizes the same objective but with a different loss function $l(\vec{u}_i, \vec{d}) = -\frac{1}{1+\exp(-\vec{u}_i^T \vec{d})}$. It however does not lead to a close form solution and has to be optimized by gradient descent algorithm.

5. EXPERIMENTS

In this section, we move forward to evaluate the effectiveness of the proposed PTE algorithm for predictive text embedding. A variety of text classification tasks and data sets are selected for this purpose. The experiments are set up as the following.

5.1 Experiment Setup

Data Sets

We select two types of text corpora, which consist of either long or short documents.

Long Document Corpora: (1) 20NG, the widely used text classification data set 20newsgroup[1], containing 20 categories; (2)Wiki, a snapshot of Wikipedia corpus in April 2010 containing around two million English articles. Only common words appeared in the vocabulary of *wiki2010* [22] are kept. We choose seven diverse categories for the classification task, including "Arts," "History," "Human," "Mathematics," "Nature," "Technology," and "Sports" from DBpedia ontology[2]. For each category, we randomly select 9,000 articles as labeled documents for training; (3) Imdb, a data set for sentiment classification from [15][3]. To avoid the distribution bias between the training and test data sets, we randomly shuffle the training and test data sets; (4) RCV1, a large benchmark corpus for text classification [12][4]. Four subsets including Corporate, Economics, Government and Market are extracted from the corpus. In RCV1 data sets, all the documents have already been represented as "bag-of-words," and orders between words are lost.

Short Document Corpora: (1) Dblp, which contains titles of papers from the computer science bibliography[5]. We choose six diverse research fields for classification including "database," "artificial intelligence," "hardware," "system," "programming languages," and "theory." For each field, we select representative conferences and collect the papers published in the selected conferences as the labeled documents; (2) Mr, a movie review data set, in which each review only contains one sentence [19][6]; (3) Twitter, a corpus

[1]Available at http://qwone.com/~jason/20Newsgroups/

[2]Available at http://downloads.dbpedia.org/3.9/en/article_categories_en.nq.bz2.

[3]Available at http://ai.stanford.edu/~amaas/data/sentiment/

[4]Available at http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm

[5]Available at http://arnetminer.org/billboard/citation

[6]Available at http://www.cs.cornell.edu/people/pabo/movie-review-data/

of Tweets for sentiment classification[7], from which we randomly sampled 1,200,000 Tweets and split them into training and testing sets.

No further text normalization such as removing stop words or stemming is done on top of the original data. We summarize the detailed statistics of these data sets in Table 1.

Compared Algorithms

We compare the PTE algorithm with other representation learning algorithms for text data, including the classical "bag-of-words" representation and the state-of-the-art approaches to unsupervised and supervised text embedding.

- BOW: the classical "bag-of-words" representation. Each document is represented with a $|V|$-dimensional vector, in which the weight of each dimension is calculated with the TFIDF weighting [21].
- Skip-gram: the state-of-the-art word embedding model proposed by Mikolov et al. [18]. For the document embedding, we simply take the average of the word embeddings as explained in Section 4.3.
- PVDBOW: the distributed bag-of-words version of paragraph vector model proposed by Le and Mikolv [10], in which the orders of words in a document are ignored.
- PVDM: the distributed memory version of paragraph vector which considers the order of the words [10].
- LINE: the large-scale information network embedding model proposed by Tang et al. [27]. We use the LINE model to learn unsupervised embeddings with the word-word network, word-document network or the combination of the two networks.
- CNN: the supervised text embedding approach based on a convolutional neural network [8]. Though CNN is proposed for modeling sentences, we adapt it for general word sequences including long documents. Although CNN typically works with fully labeled documents, it can also utilize unlabeled data by pre-training the model with unsupervised word embeddings, which is marked as CNN(pretrain).
- PTE: our proposed approach for learning predictive text embedding. There are different variants of PTE that use different combinations of the word-word, word-document and word-label networks. We denote PTE(G_{wl}) for the version that uses the word-label network only; PTE(pretrain) learns an unsupervised embedding with the word-word and word-document networks, and then fine-tune the word embeddings with the word-label network; PTE(joint) jointly trains the heterogeneous text network composed of all the three networks.

Classification and Parameter Settings

Once the vector representations of documents are constructed or learned, we apply the same classification process using the same training data set. In particular, all the documents in the training set are used in both the representation learning phase and the classifier learning phase. The class labels of these documents are not used in the representation learning phase if an unsupervised embedding method is used; they only kick in at the classifier learning phase. The class label are used in both the representation learning phase and the classifier learning phase if a predictive embedding method is used. The test data is held-out from both phases. In the classification phase, we use the one-vs-rest logistic regression model in the LibLinear package[8]. The classification performance is measured with the micro-F1 and macro-F1 metrics. For Skip-gram, PVDBOW, PVDM and PTE, the mini-batch size of the stochastic gradient descent is set as 1; the learning rate is set as $\rho_t = \rho_0(1-t/T)$, in which T is the total number of mini-batches or edge samples and $\rho_0 = 0.025$; the number of negative samples is set as 5; the window size is set as 5 in Skip-gram, PVDBOW, PVDM and when constructing the word-word co-occurrence network. We use the structure of the CNN in [8], which uses one convolution layer, followed by a max-pooling layer and a fully-connected layer. Following [8], we set the window size in the convolution layer as 3 and the number of feature maps as 100. For CNN, 1% of the training data set is randomly selected as the validation data for early stopping. The dimensionality of word vectors is set as 100 by default for all the embedding models.

Note that for the PTE models, the parameters are all set as above by default on different data sets. The only parameter that needs to be tuned is the number of samples T in edge sampling, which can be safely set to be large.

5.2 Quantitative Results

5.2.1 *Performance on Long Documents*

Table 2 and 3 compare the performance of text classification on long documents. Let us start with Table 2 on 20NG, WIKI and IMDB data sets. We first compare the performance of unsupervised embedding methods, which use either local word co-occurrences (Skip-gram, LINE(G_{ww})), document level word co-occurrences (PV-DBOW, LINE(G_{wd})), or the combination (LINE(G_{ww}+G_{wd})). We can see that the LINE(G_{wd}) with document-level word co-occurrences performs the best among the unsupervised embeddings. The performance of PVDM is inferior to that of PVDBOW, which is different from what is reported in [10]. Unfortunately we are not able to replicate their results. Similar results to ours are also reported in [14]. For the results of PVDBOW on the IMDB data set, our results are different from those reported in [10, 16]. This is because their embeddings are trained on the mixture of training and test data sets while our embeddings are only trained with the training data, which we believe is a more reasonable experiment setup.

Next we compare the performance of predictive embeddings, including different variants of CNN and PTE. When PTE is jointly trained using the heterogeneous text network or with the combination of word-document and word-label networks, it performs the best among all the approaches. All PTE approaches jointly trained with the word-label network (e.g., $G_{ww} + G_{wl}$) outperform their corresponding unsupervised embedding approaches (e.g., G_{ww}), which shows the power of learning predictive text embeddings with the supervision. PTE(joint) consistently outperforms PTE(G_{wl}), demonstrating that incorporating unlabeled information, i.e., word-word and word-document networks, also improves the quality of the embeddings. PTE(joint) also significantly outperforms PTE(pretrain). This shows that jointly training

[7]Available at http://thinknook.com/twitter-sentiment-analysis-training-corpus-dataset-2012-09-22/

[8]http://www.csie.ntu.edu.tw/~cjlin/liblinear/

Table 1: Statistics of the Data Sets

	Long Documents							**Short Documents**		
Name	20NG	WIKI	IMDB	CORPORATE	ECONOMICS	GOVERNMENT	MARKET	DBLP	MR	TWITTER
Train	11,314	1,911,617*	25,000	245,650	77,242	138,990	132,040	61,479	7,108	800,000
Test	7,532	21,000	25,000	122,827	38,623	69,496	66,020	20,000	3,554	400,000
|V|	89,039	913,881	71,381	141,740	65,254	139,960	64,049	22,270	17,376	405,994
Doc. length	305.77	672.56	231.65	102.23	145.10	169.07	119.83	9.51	22.02	14.36
#classes	20	7	2	18	10	23	4	6	2	2

*In the WIKI data set, only 42,000 documents are labeled.

Table 2: Results of text classification on **long** documents.

		20NG		Wikipedia		IMDB	
Type	Algorithm	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
Word	BOW	80.88	79.30	79.95	80.03	86.54	86.54
Unsupervised Embedding	Skip-gram	70.62	68.99	75.80	75.77	85.34	85.34
	PVDBOW	75.13	73.48	76.68	76.75	86.76	86.76
	PVDM	61.03	56.46	72.96	72.76	82.33	82.33
	LINE(G_{ww})	72.78	70.95	77.72	77.72	86.16	86.16
	LINE(G_{wd})	79.73	78.40	80.14	80.13	89.14	89.14
	LINE($G_{ww} + G_{wd}$)	78.74	77.39	79.91	79.94	89.07	89.07
Predictive Embedding	CNN	78.85	78.29	79.72	79.77	86.15	86.15
	CNN(pretrain)	80.15	79.43	79.25	79.32	89.00	89.00
	PTE(G_{wl})	82.70	81.97	79.00	79.02	85.98	85.98
	PTE($G_{ww} + G_{wl}$)	83.90	83.11	81.65	81.62	89.14	89.14
	PTE($G_{wd} + G_{wl}$)	**84.39**	**83.64**	82.29	82.27	89.76	89.76
	PTE(pretrain)	82.86	82.12	79.18	79.21	86.28	86.28
	PTE(joint)	84.20	83.39	**82.51**	**82.49**	**89.80**	**89.80**

Table 3: Results of text classification on **long** documents (RCV1 data sets).

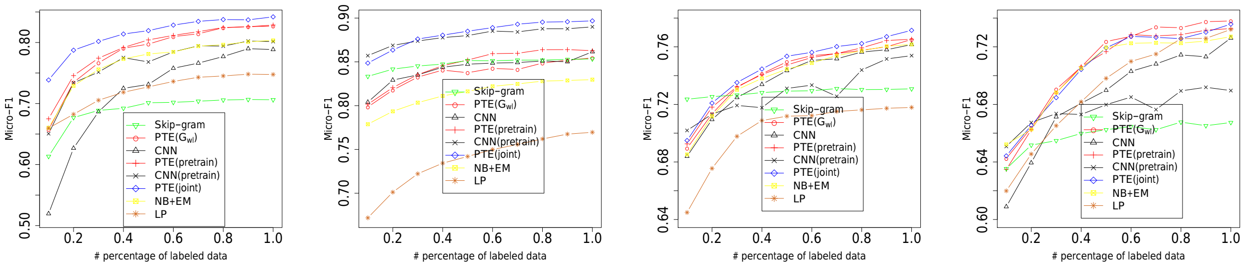
	Corporate		Economics		Government		Market	
Algorithm	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
BOW	78.45	63.80	86.18	81.67	77.43	62.38	95.55	94.09
PVDBOW	65.87	45.78	79.63	74.82	70.74	54.08	91.81	88.88
LINE(G_{wd})	76.76	60.30	85.55	81.46	77.82	63.34	95.66	93.90
PTE(G_{wl})	76.69	60.48	84.88	80.02	78.26	63.69	95.58	93.84
PTE(pretrain)	77.03	61.03	84.95	80.63	78.48	64.50	95.54	93.79
PTE(joint)	**79.20**	**64.29**	**87.05**	**83.01**	**79.63**	**66.15**	**96.19**	**94.58**

Table 4: Results of text classification on **short** documents.

		DBLP		MR		Twitter	
Type	Algorithm	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
Word	BOW	75.28	71.59	71.90	71.90	75.27	75.27
Unsupervised Embedding	Skip-gram	73.08	68.92	67.05	67.05	73.02	73.00
	PVDBOW	67.19	62.46	67.78	67.78	71.29	71.18
	PVDM	37.11	34.38	58.22	58.17	70.75	70.73
	LINE(G_{ww})	73.98	69.92	71.07	71.06	73.19	73.18
	LINE(G_{wd})	71.50	67.23	69.25	69.24	73.19	73.19
	LINE($G_{ww} + G_{wd}$)	74.22	70.12	71.13	71.12	73.84	73.84
Predictive Embedding	CNN	76.16	73.08	72.71	72.69	**75.97**	**75.96**
	CNN(pretrain)	75.39	72.28	68.96	68.87	75.92	75.92
	PTE(G_{wl})	76.45	72.74	73.44	73.42	73.92	73.91
	PTE($G_{ww} + G_{wl}$)	76.80	73.28	72.93	72.92	74.93	74.92
	PTE($G_{wd} + G_{wl}$)	**77.46**	**74.03**	73.13	73.11	75.61	75.61
	PTE(pretrain)	76.53	72.94	73.27	73.24	73.79	73.79
	PTE(joint)	77.15	73.61	**73.58**	**73.57**	75.21	75.21

with unlabeled and labeled data is much more effective compared to separating them into two phases of pre-training and fine-tuning.

It is interesting to observe that PTE(joint) consistently outperforms CNN. This is promising as PTE does not use a sophisticated neural network architecture. We also attempt to pre-train the CNN with the word embeddings learned by



(a) 20NG (b) IMDB (c) DBLP (d) MR

Figure 2: Performance w.r.t. # labeled data.

the LINE($G_{wl} + G_{wd}$) and then fine tune it with the labels. Surprisingly, the performance of CNN with pre-training significantly improves on the 20NG and IMDB data sets and remains almost the same on the WIKI data set. This implies that pre-training CNN with a well learned unsupervised embeddings can be very useful. However, even with pre-training the performance of CNN is still inferior to that of PTE(joint). This is probably because the PTE model can jointly train with both the unlabeled and labeled data while CNN can only utilize them separately through pre-training and fine-tuning. PTE(joint) also outperforms the classical "bag-of-words" representation even though the dimensionality of the embeddings learned by the PTE is way smaller than that of "bag-of-words."

Table 3 reports the results on the RCV1 data sets. As the order between the words is lost, the embedding methods that require the word order information are not applicable. Similar results are observed. Predictive text embeddings outperform unsupervised embeddings. PTE(joint) is also much more effective than PTE(pretrain).

All the embedding approaches (except "bag-of-words") are trained with asynchronous stochastic gradient descent algorithm using 20 threads on a single machine with 1T memory, 40 CPU cores at 2.0GHZ. We compare the running time of CNN and PTE(joint) on the IMDB data set. The PTE(joint) method is typically more than 10 times faster than the CNN models. When pre-trained with preexisting word embeddings, CNN converges much faster, but still more than 5 times slower than PTE(joint).

5.2.2 *Performance on Short Documents*

Table 4 compares the performance on short documents. Among unsupervised embeddings, the LINE($G_{ww} + G_{wd}$), which combines the document-level and local context-level word co-occurrences, performs the best. The LINE(G_{ww}) utilizing the local context-level word co-occurrences outperforms the LINE(G_{wd}) using document-level word co-occurrences, which is opposite to the observations on long documents. This is because document-level word co-occurrences suffer from the sparsity in short documents, with similar results observed in statistical topic models [26]. The performance of PVDM is still inferior to that of PVDBOW, which is consistent with the results on long documents.

For predictive embeddings, the best performance is obtained by the PTE (on DBLP, MR) or CNN (on TWITTER). Among the PTE approaches, the predictive embeddings learned by incorporating the word-label network outperform the corresponding unsupervised embeddings, which is consistent with the results on long documents. PTE(joint) outperforms LINE(G_{wl}), showing the usefulness of incorporating unlabeled information. PTE(joint) also significantly outperforms the PTE(pretrain), showing the advantage of jointly training with the labeled and unlabeled data.

On the short documents, we observe that PTE(joint) does not consistently outperform the CNN. The reason is probably due to the problem of word sense ambiguity, which becomes more serious on the short documents. CNN reduces the problem of word ambiguity through using the word orders in local context in the convolutional kernels while PTE does not leverage the orders. We believe there is considerable room to improve predictive text embedding by utilizing word orders, which we leave as future work.

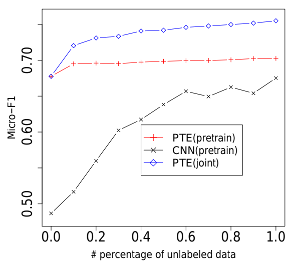
5.3 Effects of Labeled Data

We compare CNN and PTE head-to-head by varying the percentages of labeled data. We consider the cases without or with unlabeled data, mimicking the scenarios of supervised and semi-supervised learning. In the setting of semi-supervised learning, we also compare with classical semi-supervised approaches, Naive Bayes with EM (NB+EM) [25] and label propagation (LP) [31]. Fig. 2 reports the performance on both long and short documents. Overall, both CNNs and PTEs improve when the size of labeled data increases. In the supervised settings, i.e., between CNN and PTE(G_{wl}), PTE(G_{wl}) outperforms or is comparable to CNN on both the long and short documents. In the semi-supervised settings, i.e., between CNN(pretrain) and PTE(joint), PTE(joint) consistently outperforms CNN(pretrain), which is pre-trained with the **best performing** unsupervised word embeddings. The PTE(joint) also outperforms the state-of-the-art semi-supervised approaches Naive Bayes with EM and label propagation.

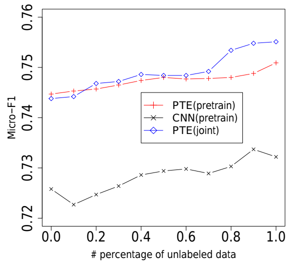
We also notice that when the size of labeled data is scarce, pre-training CNN with unsupervised embeddings is quite helpful, especially on the short documents. It even outperforms all PTEs when the training examples are too few. However, when the size of labeled data increases, pre-training CNN does not always improve its performance (e.g., on the DBLP and MR data sets).

Note that for Skip-gram, increasing the number of labeled data in training does not further increase the performance. We also notice that when the labeled documents are too few, the performance of PTE is inferior to the Skip-gram on the DBLP data set. The reason is that when the number of labeled examples is scarce, the word-label network is very noisy and PTE treats the word-label network equally to the robust word-word/word-document networks. A way to fix is to adjust the sampling probability from the word-label and word-word/word-document when the labeled data is scarce. We leave it as future work.

5.4 Effects of Unlabeled Data



(a) 20NG

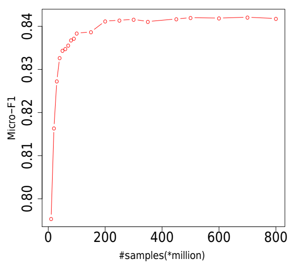


(b) DBLP

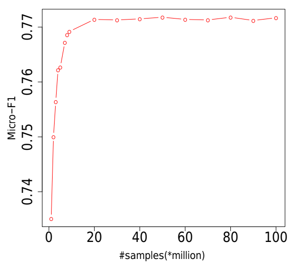
Figure 3: Performance w.r.t. # unlabeled data.

We also analyze the performance of the CNNs and PTEs w.r.t. the size of unlabeled data. For CNN, the unlabeled data is used for pre-training while for the PTE, the unlabeled data can be used for either pre-training or jointly training. Fig. 3 reports the results on 20NG and DBLP data sets. Due to space limitation, we omit the results on other data sets, which are similar. On 20NG, we use 10% documents as labeled while the rest is used as the unlabeled; on DBLP, we randomly sample 200,000 titles of the papers published in the other conferences as the unlabeled data. We use the unsupervised embeddings (learned by LINE($G_{ww} + G_{wd}$)) as the pre-training of CNN. We can see that the performance of both CNN and PTE improves when the size of unlabeled data increases. For PTE, the way of jointly training with the unlabeled and labeled data is much more effective than separating them into pre-training and fine-tuning.

5.5 Parameter Sensitivity



(a) 20NG



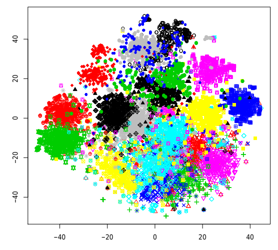
(b) DBLP

Figure 4: Sensitivity of PTE w.r.t. number of samples T.

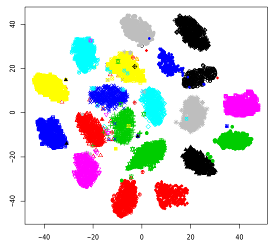
For the proposed PTE models, as mentioned previously, most of the parameters except for the number of edge samples T are not sensitive to different data sets and can be set by default. Here we analyze the performance sensitivity of PTE(joint) w.r.t the number of samples T. Fig. 4 reports the results on the 20NG and DBLP data sets. On both data sets, we can see that when the number of samples T becomes large enough, the performance of PTE(joint) converges. Therefore, in practice, one can just set the number of samples T to be sufficiently large. A reasonable estimation of T we find in practice is several times of the number of edges in the heterogeneous text networks.

5.6 Document Visualization

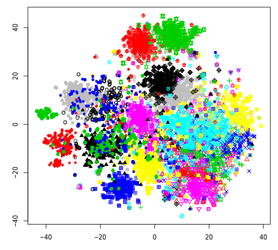
Finally, we give an illustrative visualization of the documents in 20NG to compare the unsupervised and predictive



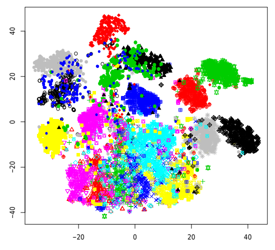
(a) Train(LINE(G_{wd}))



(b) Train(PTE(G_{wl}))



(c) Test(LINE(G_{wd}))



(d) Test(PTE(G_{wl}))

Figure 5: Document visualization using unsupervised and predictive embeddings on 20NG data set, visualized with the t-SNE tool [29].

embeddings. We used LINE(G_{wd}) for the unsupervised embedding and PTE(G_{wl}) for the predictive embedding. Both the training and test documents are visualized. Fig. 5 shows the visualization results. We can see that on both training and test data sets, the predictive embedding much better distinguishes different classes than those learned by the unsupervised embedding, which intuitively shows the power of predictive embeddings on the task of text classification.

6. DISCUSSION AND CONCLUSION

Unsupervised embeddings. The essential information used by the unsupervised approaches is the local context-level or document-level word co-occurrences. On long documents, we can see that document-level word co-occurrences are more useful than the local context-level word co-occurrences, and the combination of two does not further improve the result; on short documents, the local context-level word co-occurrences are more useful than the document-level word co-occurrences, and their combination will further improve the embedding. Document-level word co-occurrences suffers from short lengths of documents.

Predictive embeddings. Comparing between CNN and PTE, the CNN model seems to handle labeled information more effectively, especially on short documents. This is because CNN uses a much more complicated structure than the PTE, which in particular utilizes word orders in the local context and addresses word sense ambiguity.

Therefore, in the case when labeled data is very sparse, CNN can outperform the PTE, especially on short documents. However, this advantage is at the expense of intensive computation and exhaustive parameter tuning. On the other hand PTE is much faster and much easier to configure (with few parameters to tune). When the labeled data becomes abundant, the performance of PTE will be at least comparable and usually superior to CNN.

Compared to the CNN model, an obvious advantage of the PTE model is its capability of being able to jointly train with both the unlabeled and labeled data. CNN can only make use of unlabeled data through an indirect way, i.e., pre-training with the unsupervised word embeddings learned from other methods. Pre-training may not always help when the size of labeled data becomes abundant.

Practical Guidelines.

Based on the above discussions, we provide the following practical guidelines, which suggest to choose between CNN or PTE in different scenarios.

(1) When no labeled data is available, we suggest using LINE(G_{wd}) for learning an unsupervised word embedding from long documents and using LINE($G_{wd}+G_{ww}$) from short documents.

(2) When few labeled data is available, on short documents, we suggest to learn an unsupervised embedding first (according to the first guideline) and then pre-train CNN with the unsupervised word embedding; on long documents, we suggest using PTE.

(3) When the labeled data are abundant, on long documents, we strongly suggest using the PTE model to jointly train the embedding with both the unlabeled and labeled data; on short documents, the selection between PTE(joint) and CNN or CNN(pretrain) basically trades performance with efficiency.

We believe this study provides an interesting direction to efficiently learn predictive, distributed text embeddings. It is feasible and desirable to develop similar methods that compete with deep neural networks end-to-end on specific tasks but avoid complex model architectures or intensive computation. There is considerable room to improve PTE, for example by considering the orders of words.

Acknowledgments

Qiaozhu Mei is supported by the National Science Foundation under grant numbers IIS-1054199 and CCF-1048168.

7. REFERENCES

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591, 2001.

[2] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[3] S. Bhagat, G. Cormode, and S. Muthukrishnan. Node classification in social networks. In *Social network data analytics*, pages 115–148. Springer, 2011.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 3:993–1022, 2003.

[5] P. Blunsom, E. Grefenstette, N. Kalchbrenner, et al. A convolutional neural network for modelling sentences. In *ACL*, 2014.

[6] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537, 2011.

[7] J. R. Firth. A synopsis of linguistic theory, 1930–1955. *In J. R. Firth (Ed.), Studies in linguistic analysis*, pages 1–32.

[8] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[9] J. B. Kruskal and M. Wish. *Multidimensional scaling*, volume 11. Sage, 1978.

[10] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

[11] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361:310, 1995.

[12] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 5:361–397, 2004.

[13] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[14] Y. Liu, Z. Liu, T.-S. Chua, and M. Sun. Topical word embeddings. 2015.

[15] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *ACL-HLT*, pages 142–150, 2011.

[16] G. Mesnil, M. Ranzato, T. Mikolov, and Y. Bengio. Ensemble of generative and discriminative techniques for sentiment analysis of movie reviews. *arXiv preprint arXiv:1412.5335*, 2014.

[17] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048, 2010.

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[19] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124, 2005.

[20] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *KDD*, pages 701–710. ACM, 2014.

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

[22] C. Shaoul. The westbury lab wikipedia corpus. *Edmonton, AB: University of Alberta*, 2010.

[23] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *CIKM*, pages 101–110. ACM, 2014.

[24] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, volume 1631, page 1642, 2013.

[25] J. Su, J. S. Shirab, and S. Matwin. Large scale text classification using semi-supervised multinomial naive bayes. In *ICML*, pages 97–104, 2011.

[26] J. Tang, Z. Meng, X. Nguyen, Q. Mei, and M. Zhang. Understanding the limiting factors of topic modeling via posterior contraction analysis. In *ICML*, pages 190–198, 2014.

[27] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077, 2015.

[28] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[29] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9(2579-2605):85, 2008.

[30] J. Weston, S. Chopra, and K. Adams. tagspace: Semantic embeddings from hashtags. In *EMNLP*, pages 1822–1827, 2014.

[31] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *NIPS*, 16(16):321–328, 2004.