Deep Semantic Embedding

Hao Wu[*]
Department of Computer Science
University of Southern California
hwu732@usc.edu

Martin Renqiang Min
Department of Machine Learning
NEC Labs America
renqiang@nec-labs.com

Bing Bai
Department of Machine Learning
NEC Labs America
bbai@nec-labs.com

ABSTRACT

We introduce Deep Semantic Embedding (DSE), a supervised learning algorithm which computes semantic representation for text documents by respecting their similarity to a given query. Unlike other methods that use single-layer learning machines, DSE maps word inputs into a low-dimensional semantic space with deep neural network, and achieves a highly nonlinear embedding to model the human perception of text semantics. Through discriminative fine-tuning of the deep neural network, DSE is able to encode the relative similarity between relevant/irrelevant document pairs in training data, and hence learn a reliable ranking score for a query-document pair. We present test results on datasets including scientific publications and user-generated knowledge base.

Keywords

Deep Learning, Nonlinear Embedding, Semantic Indexing, Ranking

1. INTRODUCTION

In this paper, we consider modeling underlying structure of text documents for information retrieval. The goal is to find concise representation of text semantics while preserving discriminative features that are useful for similarity judgment in ranking documents with respect to a query.

Modern Internet search is built on the fundamental task of ranking documents from database and returning relevant ones to a given query. The quality of document ranking largely depends on measuring the semantic similarity of query-document pairs. Typically, semantic similarity can be measured in terms of occurrences of "words" or "terms". For example, the popular TFIDF [19] scheme represents each document using frequency count of words in a basic vocabulary, and normalize the values using inverse document frequency count. In ranking tasks, the relevance score is usually measured by similarity metrics, typically cosine similarity between query-document pairs in terms of TFIDF vector representation.

[*]Most of this work was done when the author was an intern at NEC Labs America.

While the TFIDF scheme has been successfully adopted in practical search engines due to its simplicity and efficiency, the approach reveals little statistical structure in text corpus, e.g., the occurrences of words within or between documents. The exact match of words ignores the similarity between synonyms and cannot distinguish polysemy. To address these issues, directed graphical models, notably probabilistic Latent Semantic Indexing (pLSI [11]), as a variant of Latent Semantic Indexing (LSI [6]) and Latent Dirichlet Allocation (LDA [3]) are proposed. pLSI and LDA model each document as mixing proportions for latent components which are viewed as "topics", and a topic is represented as a probabilistic distribution over words to capture their correlations.

Recent approaches [18, 12, 22] using undirected graphical models are based on Restricted Boltzmann Machines (RBMs). RBMs factorize data distributions into a particular form of the Product of Experts (PoE) rather than a mixture of latent aspects, and have a complementary prior over hidden units to solve the "explaining-away" problem of directed graphical models. These models hence generally outperform pLSI and LDA.

However, all the above-mentioned models consider little *supervised* information for ranking documents in search tasks. The supervised information could come from human labeling, or more abundant in the form of implicit relevance feedback, e.g., in query log where the web documents clicked-through by users can be deemed to be more relevant to a query than the others. Recent models attempt to use the supervised signal to discriminatively train a mapping from the word content in a query-document pair to a relevance score. Supervised Semantic Indexing (SSI [1]) and its variant Polynomial Semantic Indexing (PSI [2]) using extended high-order word features fall into this research category. Despite having a *nonlinear* function to calculate a similarity score from word features in a query-document pair, the low-rank approximation makes the models boil down to finding the *linear* mapping from word content to a latent space, in which the similarity of a query-document pair is computed. Nevertheless, the *linear* word embeddings are not powerful enough to capture the profound semantics in text documents. It has been shown that deep nonlinear feature embedding can significantly improve classification performance in other tasks [16, 15], so we expect that deep nonlinear feature mapping will help document ranking similarly.

In this paper, we consider learning *nonlinear* semantic embeddings from word content. We present a model called Deep Semantic Embedding (DSE), which exploits the nonlinearity between word features and latent semantics with Deep Neural Network (DNN) composed of stacked RBMs [9]. With word inputs, the activation of hidden units on the top layer of the DNN serves as the coordinate of a document in the latent semantic space. By learning from the relative similarity between relevant/irrelevant document pairs in training data, DSE is able to preserve the discriminative features for computing a reliable ranking score of a query-document pair. For disciminative learning, we use gradient descent method to fine-tune the DNN in a back-propagation manner. To evaluate DSE, we present empirical analysis on real data including NIPS publications, and Wipipedia web pages.

2. DEEP SEMANTIC EMBEDDING

In typical search tasks, we are interested in retrieving relevant documents from a text corpus to respond to a given query. Suppose we represent a query using vector $\mathbf{q} \in \mathbb{R}^D$, and the set of N documents in the corpus as $\mathcal{D} = \{\mathbf{d}^{(i)}\}_{i=1}^N \subset \mathbb{R}^D$, where D is the vocabulary size of words. The choices of vector representation can be term frequency, or its variants such as TFIDF, or binary occurrence of words. We use q_j and d_j to denote the j^{th} feature dimension of a query $\mathbf{q}$ and a document $\mathbf{d}$ respectively.

2.1 The Algorithm

Given observation $\mathbf{v}$ which denotes input $\mathbf{q}$ or $\mathbf{d}$, we consider nonlinear models $\Phi : \mathbb{R}^D \mapsto \mathbb{R}^C$ which map word features into a latent semantic space and output C dimensional embedding $\mathbf{h} = \Phi(\mathbf{v})$. Here, $C < D$ in typical cases of dimension reduction. It is worthy to notice $\mathbf{q}$ and $\mathbf{d}$ are heterogeneous in real data. Queries are usually much shorter and have different word distributions, compared to documents. However, for simplicity, we deem a query as a special type of document and model $\mathbf{q}$ and $\mathbf{d}$ in a uniform way using the same $\Phi(\cdot)$. This is typically useful in the setting of document-document retrieval. One may also easily generalize our model to common cases of query-document retrieval, e.g., use different $\Phi(\cdot)$ for $\mathbf{q}$ and $\mathbf{d}$ respectively.

To model the degree of relevance between $\mathbf{q}$ and $\mathbf{d}$, we compute the similarity of their latent embeddings using the following bilinear model:

$$s(\mathbf{q}, \mathbf{d}) = \Phi(\mathbf{q})^\top \mathbf{A} \Phi(\mathbf{d}) \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{C \times C}$ is the weight matrix to be learned. Without loss of generality, we would like the learning power of $\mathbf{A}$ to be absorbed by $\Phi(\cdot)$, and set $\mathbf{A}$ as identity matrix $\mathbf{I}$. The model hence can be simplified as

$$s(\mathbf{q}, \mathbf{d}) = \Phi(\mathbf{q})^\top \Phi(\mathbf{d}) \tag{2}$$

To learn $\Phi(\cdot)$, we choose a multi-layer deep neural network (DNN) pre-trained with stacked Restricted Boltzmann Machines (RBMs), which is deemed to have more representational power of learning semantics in text than a single RBM or other single-layer learning machines [13]. RBM is an undirected graphical model with visible units and hidden units bipartitely connected. A L-layer DNN with stacked RBMs takes observation $\mathbf{v} = \mathbf{h}^0$ as input in the first layer, and learns hidden units $\mathbf{h}^k$ for $k = 1, 2, \cdots, L$ in each of L layers respectively. The word features pass through DNN and get digested layer by layer to represent semantics in this process. This is to simulate the way of human interpretation of semantics in a document, where DNN models the function of human brain, as there is intriguing connection between them [9]. The learned hidden units $\mathbf{h}^L$ in the top layer is used as the final semantic representation, that is $\Phi(\mathbf{v}) = \mathbf{h}^L$.

2.2 Restricted Boltzmann Machines

During the pre-training of the DNN as stacked RBMs, each RBM takes $\mathbf{h}^{k-1}$ as input, and output $\mathbf{h}^k$ which is further used as input for layer $k+1$. The joint probability $p(\mathbf{h}^{k-1}, \mathbf{h}^k)$ of RBM in each layer is defined as

$$p(\mathbf{h}^{k-1}, \mathbf{h}^k) = \frac{1}{Z} exp(-E(\mathbf{h}^{k-1}, \mathbf{h}^k)) \tag{3}$$

where Z is a partition function defined as the sum of $exp(-E(\mathbf{h}^{k-1}, \mathbf{h}^k)$ over all possible configurations. With stochastic binary units modeling both the input and the output in RBM, the energy term $E(\mathbf{h}^{k-1}, \mathbf{h}^k)$ is defined as

$$E(\mathbf{h}^{k-1}, \mathbf{h}^k) = -\sum_{ij} W_{ij}^k h_i^{k-1} h_j^k - \sum_i h_i^{k-1} b_i^k - \sum_j h_j^k c_j^k \tag{4}$$

where $\mathbf{W}^k$ is the weight matrix for layer k, and $\mathbf{b}^k$ and $\mathbf{c}^k$ are the bias vectors. Given the hidden states $\mathbf{h}^k$ in layer k, the states h_j^{k-1} are conditionally independent. We have $p(\mathbf{h}^{k-1}|\mathbf{h}^k) = \prod_i p(h_i^{k-1}|\mathbf{h}^k)$, and

$$p(h_i^{k-1} = 1|\mathbf{h}^k) = \sigma(b_i^k + \sum_j W_{ij}^k h_j^k) \tag{5}$$

where $\sigma(x) = \frac{1}{1+exp(-x)}$ is the sigmoid function. $p(\mathbf{h}^k|\mathbf{h}^{k-1})$ has the similar form $p(\mathbf{h}^k|\mathbf{h}^{k-1}) = \prod_j p(h_j^k|\mathbf{h}^{k-1})$, and the conditional probabilities $p(h_j^k = 1|\mathbf{h}^{k-1})$ are expressed as

$$p(h_j^k = 1|\mathbf{h}^{k-1}) = \sigma(c_j^k + \sum_i W_{ij}^k h_i^{k-1}) \tag{6}$$

We can also model output $\mathbf{h}^k$ as Gaussian latent variables [23]. This unsupervised model represents an undirected alternative of pLSI [11]. The energy function is given by:

$$E(\mathbf{h}^{k-1}, \mathbf{h}^k) = -\sum_{ij} W_{ij}^k h_i^{k-1} \frac{h_j^k}{\sigma_j^k} - \sum_i h_i^{k-1} b_i^k - \sum_j \frac{(h_j^k - c_j^k)^2}{2(\sigma_j^k)^2} \tag{7}$$

where σ_j^k is the variance of h_j^k. And the conditional probabilities can be rewritten as:

$$p(h_i^{k-1} = 1|\mathbf{h}^k) = \sigma(b_i^k + \sum_j W_{ij}^k h_j^k) \tag{8}$$

$$p(h_j^k = h|\mathbf{h}^{k-1}) = \mathcal{N}(h, c_j^k + \sigma_j^k \sum_i W_{ij}^k h_i^{k-1}, \sigma_j^k) \tag{9}$$

where $\mathcal{N}(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(x-\mu)^2}{2\sigma^2})$ is a Gaussian distribution with mean μ and variance σ.

The marginal distribution over $\mathbf{h}^{k-1}$ is given by

$$p(\mathbf{h}^{k-1}) = \frac{1}{Z}\sum_{\mathbf{h}^k} exp(-E(\mathbf{h}^{k-1}, \mathbf{h}^k)) \quad (10)$$

To learn the parameters $\mathbf{\Theta} = \{\mathbf{W}^k\}_{k=1}^L$, we use the greedy procedure proposed in [8] to train the DNN in a bottom-up layer-wise manner. Gradient ascent is used to maximize the log-likelihood of $p(\mathbf{h}^{k-1})$ and we get the gradients $\frac{\partial logp(\mathbf{h}^{k-1})}{\partial W_{ij}^k}$. For binary $\mathbf{h}^k$, the parameter updates are given by:

$$\Delta W_{ij}^k = \epsilon(< h_i^{k-1}h_j^k >_{data} - < h_i^{k-1}h_j^k >_{model}) \quad (11)$$

were ϵ is the learning rate, $< \cdot >_{data}$ defines the expectation with respect to the data distribution and $< \cdot >_{model}$ is the expectation with respect to the distribution defined by the model. For Gaussian $\mathbf{h}^k$, the updates become:

$$\Delta W_{ij}^k = \epsilon(< h_i^{k-1}\frac{h_j^k}{\sigma_j^k} >_{data} - < h_i^{k-1}\frac{h_j^k}{\sigma_j^k} >_{model}) \quad (12)$$

In practice, we fix variance at $(\sigma_j^k)^2 = 1$ for all units h_j^k, and in this case Eq. (12) becomes the same as defined in Eq. (11).

To avoid computing $< \cdot >_{model}$ which cannot be achieved analytically in less than exponential time, contrastive divergence can be used [7]:

$$\Delta W_{ij}^k = \epsilon(< h_i^{k-1}h_j^k >_{data} - < h_i^{k-1}h_j^k >_T) \quad (13)$$

where we run T steps Gibbs sampling to approximate the expectation with respect to model distribution. In practice, large values of T are seldom needed and even $T = 1$ can approximate maximum likelihood learning well [8].

2.3 Discriminative Fine-tuning

While using RBMs [18, 12, 22] in deep architecture can exploit the learning power of generative models, there is little supervised information used. In real search settings, we may consider human labeling of relevant/irrelevant documents with respect to a query. However, it maybe expensive to obtain label information. Search query logs are another valuable resource which is abundantly available. Typically, there are user feedback information in search sessions recorded. We can deem the web documents that were clicked-through by users are more relevant than those were not.

Formally, we consider training data in the form of tuples $t = (\mathbf{q}, \mathbf{d}^+, \mathbf{d}^-)$, where $\mathbf{d}^+$ is a relevant document with respect to a query $\mathbf{q}$, and $\mathbf{d}^-$ is an irrelevant document. After training the DNN with RBMs, we have $\Phi(\cdot)$ pre-trained. In this discriminative learning stage, we impose the ranking score $s(\mathbf{q}, \mathbf{d}^+)$ should be sufficiently larger than $s(\mathbf{q}, \mathbf{d}^-)$ so that $\mathbf{d}^+$ would be ranked higher than $\mathbf{d}^-$. We penalize those tuples that are not ranked in the correct order, and define the margin ranking loss function [20] as:

$$F = \sum_t f(t) = \sum_{t=(\mathbf{q},\mathbf{d}^+,\mathbf{d}^-)} max(0, 1 - s(\mathbf{q}, \mathbf{d}^+) + s(\mathbf{q}, \mathbf{d}^-)), \quad (14)$$

where $max(0, x)$ defines the hinge loss function. The optimal parameters $\mathbf{\Theta}^* = \{\mathbf{W}^k\}_{k=1}^L$ should minimize F. To learn the model, we use stochastic gradient descent. Given a random sample t, the procedure updates $\mathbf{W}^k$ using

$$\Delta W_{ij}^k = \epsilon\frac{\partial f(t)}{\partial W_{ij}^k} \quad (15)$$

If the sample t yields $1 - s(\mathbf{q}, \mathbf{d}^+) + s(\mathbf{q}, \mathbf{d}^-) <= 0$, $f(t) = 0$ and updating the weight matrices is not needed. But if it results in a margin violation that $1 - s(\mathbf{q}, \mathbf{d}^+) + s(\mathbf{q}, \mathbf{d}^-) > 0$, we have to update the weights. In the case of margin violation, we may rewrite $f(t)$ as

$$f(t) = 1 - s(\mathbf{q}, \mathbf{d}^+) + s(\mathbf{q}, \mathbf{d}^-) \quad (16)$$

Substituting $s(\mathbf{q}, \mathbf{d}^+)$ and $s(\mathbf{q}, \mathbf{d}^-)$ with Eq. (2), we have

$$f(t) = 1 - \Phi(\mathbf{q})^\top\Phi(\mathbf{d}^+) + \Phi(\mathbf{q})^\top\Phi(\mathbf{d}^-) \quad (17)$$

We update $\{\mathbf{W}^k\}_{k=1}^L$ layer by layer in top-down manner. Let $\mathbf{h}^{L(\cdot)} = \Phi(\cdot)$ denote the hidden representation output in top layer (i.e., layer L). Eq. (17) can be rewritten as

$$f(t) = 1 - (\mathbf{h}^{L(\mathbf{q})})^\top\mathbf{h}^{L(\mathbf{d}^+)} + (\mathbf{h}^{L(\mathbf{q})})^\top\mathbf{h}^{L(\mathbf{d}^-)} \quad (18)$$

Let $\mathbf{H}^L = [\mathbf{h}^{L(\mathbf{q})}, \mathbf{h}^{L(\mathbf{d}^+)}, \mathbf{h}^{L(\mathbf{d}^-)}]^\top$ be the representation matrix denoting the output of $t = (\mathbf{q}, \mathbf{d}^+, \mathbf{d}^-)$ on the top layer of DNN. The derivatives of $f(t)$ with respect to $\mathbf{H}^L$ is given by:

$$\frac{\partial f(t)}{\partial \mathbf{H}^L} = [\mathbf{h}^{L(\mathbf{d}^-)} - \mathbf{h}^{L(\mathbf{d}^+)}, -\mathbf{h}^{L(\mathbf{q})}, \mathbf{h}^{L(\mathbf{q})}]^\top \equiv \mathbf{\Delta}^L \quad (19)$$

We then update $\mathbf{W}^L$ in the top layer. Using chain rule, we compute the derivatives $\frac{\partial f(t)}{\partial W_{ij}^L}$

$$\frac{\partial f(t)}{\partial \mathbf{W}^L} = \sum_{mj} \Delta_{mj}^L \frac{\partial \Delta_{mj}^L}{\partial \mathbf{W}^L} \quad (20)$$

where Δ_{mj}^L is the $(m, j)^{th}$ element of $\mathbf{\Delta}^L$. Using binary units for output on the top layer, the above computation can be obtained:

$$\frac{\partial f(t)}{\partial \mathbf{W}^L} = (\mathbf{H}^{L-1})^\top(\mathbf{\Delta}^L \circ \mathbf{H}^L \circ (1 - \mathbf{H}^L)) \quad (21)$$

where $\mathbf{H}^{L-1}$ is the representation matrix for the output on layer $L - 1$, and $\circ$ denotes the Hadamard (element-wise) product. If we model the output using Gaussian units on the top layer, we have

$$\frac{\partial f(t)}{\partial \mathbf{W}^L} = (\mathbf{H}^{L-1})^\top\mathbf{\Delta}^L \quad (22)$$

The derivatives $\frac{\partial f(t)}{\partial \mathbf{W}^k}$ for $k = L-1, L-2, \cdots, 1$ are computed layer-wise in top down manner using back-propagation algorithm. We first compute the derivatives of $f(t)$ with respect to the output $\mathbf{H}^k$ on layer k

$$\frac{\partial f(t)}{\partial \mathbf{H}^k} = \sum_{mj} \Delta_{mj}^{k+1} \frac{\partial \Delta_{mj}^{k+1}}{\partial \mathbf{H}^k} \quad (23)$$

which uses the derivatives propagated from the upper layer (i.e., layer $k + 1$) $\frac{\partial f(t)}{\partial \mathbf{H}^{k+1}} \equiv \mathbf{\Delta}^{k+1}$. The computation results of the above equation are

$$\frac{\partial f(t)}{\partial \mathbf{H}^k} = (\mathbf{\Delta}^{k+1} \circ \mathbf{H}^{k+1} \circ (1 - \mathbf{H}^{k+1}))(\mathbf{W}^{k+1})^\top \equiv \mathbf{\Delta}^k \quad (24)$$

$$\frac{\partial f(t)}{\partial \mathbf{H}^k} = \mathbf{\Delta}^{k+1}(\mathbf{W}^{k+1})^\top \equiv \mathbf{\Delta}^k \tag{25}$$

for binary output and Gaussian output respectively. To compute the derivatives $\frac{\partial f(t)}{\partial \mathbf{W}^k} = \sum_{mj} \Delta^k_{mj} \frac{\partial \Delta^k_{mj}}{\partial \mathbf{W}^k}$, we may reuse Eq. (21) and Eq.(22) for a specific layer (layer k).

3. RELATIONSHIP TO OTHER MODELS

3.1 Siamese networks

The proposed model can be viewed as a Siamese network [5] with pretraining of RBM. As proved in deep learning literature, such pretraining is crucial in many cases, especially when dimensionality of the problem is large.

3.2 Supervised Semantic Indexing

Supervised Semantic Indexing (SSI [1]) learns a ranking score $s(\mathbf{q}, \mathbf{d})$ in the form

$$s(\mathbf{q}, \mathbf{d}) = \mathbf{q}^\top \mathbf{A}\mathbf{d}, \tag{26}$$

where $\mathbf{A}$ is the weight matrix. Low-rank decomposition is used to approximate $\mathbf{A}$. If we consider a symmetric decomposition in the form $\mathbf{A} = \mathbf{U}^\top\mathbf{U} + \mathbf{I}$, $s(\mathbf{q}, \mathbf{d})$ can be rewritten as:

$$s(\mathbf{q}, \mathbf{d}) = \mathbf{q}^\top(\mathbf{U}^\top\mathbf{U} + \mathbf{I})\mathbf{d} = (\mathbf{U}\mathbf{q})^\top(\mathbf{U}\mathbf{d}) + \mathbf{q}^\top\mathbf{d}. \tag{27}$$

The right term $\mathbf{q}^\top\mathbf{d}$ represents the similarity of $\mathbf{q}$ and $\mathbf{d}$ in original word space. If we use normalized TFIDF vectors as inputs, $\mathbf{q}^\top\mathbf{d}$ is equivalent to the TFIDF scheme using cosine similarity. The left term can be deemed as to firstly learn *linear* embeddings for $\mathbf{q}$ and $\mathbf{d}$ using transformation matrix $\mathbf{U}$, then measure their similarity based on the embedding. Our DSE measures similarity on embeddings in latent space as well, but the embeddings are achieved using *nonlinear* transformation $\Phi(\cdot)$ which have more learning power for model word semantics.

3.3 Deep Match

Deep match [14] is a ranking algorithm using deep neural networks. This work features a deep network structure that connections between layers are learned from a clever clustering algorithm using LDA. In comparison, our work falls on classical pretraining-finetuning style of deep learning.

3.4 Deep Ranknet

Deep ranknet [21] is an effort to extend ranknet [4] to use deep neural networks. Unlike the other works mentioned above, ranknet doesn't generated individual embeddings for a query and a document, but make features directly using both query and document (e.g BM25 similarity between query and document).

4. EXPERIMENTS

For evaluation, we compare DSE with Supervised Semantic Indexing (SSI [1]) which significantly outperforms other unsupervised algorithms such as LSI and TFIDF scheme. For that, we used the same wikipedia link prediction dataset used in [1]. We added NIPS conference dataset to look into more details of DSE, e.g., when training set is small.

It will be good to compare to a dataset that is more common in IR community. However, the benchmark set like LETOR or TREC usually only provide similarity or statistics features (e.g. BM25 for query and title of doc). The goal of our algorithm is to learn from low level features like words without much feature engineering, which will it more adaptive to new tasks, especially when the query and documents are of different modality (e.g. Image retrieval).

Also note that comparison to a VSM model using TFIDF was done in [1] that showed superiority of SSI. Since we outperform SSI, we skipped baseline of BM25, since BM25 and TFIDF are similar in principle, and in many experimental evaluations.

4.1 Parameter Setting of DSE

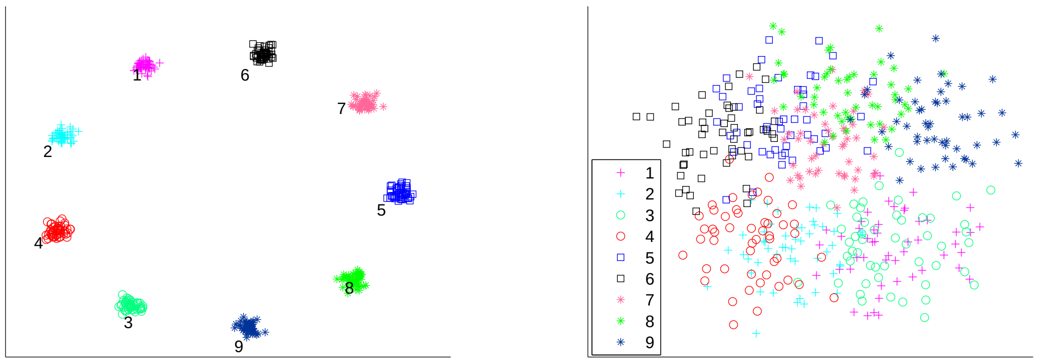
We train DSE typically with an architecture $D - \frac{1}{2}D - \frac{1}{2}D - D - C$, where D is the feature dimensionality and C the dimensionality of the resulting embeddings. We use binary units in first three layers of DSE and linear units with Gaussian noise in the top layer. The weights were initialized with small random values sampled from a normal distribution. The weights are updated using a learning rate in the range of [0.01, 1], a momentum of 0.9 to speed up convergence and a weight decay rate of 0.0001 to avoid overfitting.

4.2 NIPS Conference Papers

We first validate our method on a small data of scientific articles. We use the NIPS 0-12 dataset[1], which contains papers from the NIPS conferences between 1987 and 1999. The papers are organized into 9 sections according to different scientific fields. We randomly sample 50 papers from each section and use them for the ranking task. Each paper is deemed as relevant to another if both of them belong to the same section, and otherwise they are not relevant. We use binary features of word-document occurrence. For preprocessing, we remove the words that occur in more than 200 documents and less than 50 documents. Those words are either two common without any discriminative information or too specific (not generalize well). This results in 1082 distinct words in vocabulary, and we use top 200 of them to show more clearly the capability difference between DSE and SSI. We generate all possible tuples $(\mathbf{q}, \mathbf{d}^+, \mathbf{d}^-)$ for the dataset. To train the models, we use tuples with a portion of randomly sampled distinct positive links $(\mathbf{q}, \mathbf{d}^+)$. We first use 40% of all positive links to train DSE with a architecture $200 - 100 - 100 - 200 - C$ with $C = 2$, and SSI with same 2 dimensional embeddings as output. We visualize the resulting 2 dimensional embeddings for each model in Fig. 1. DSE can easily distinguish each section of papers in the latent semantic space, while the class boundaries in SSI are not that clear. It should be noted that as we increase the number of word features to 1082, SSI can also achieve perfect class boundary. But it is clear that DSE has better capability to model the supervised signal from a small number of input features, than a linear model like SSI.

To get a more complete picture of their performance difference, we vary the portion of positive links $(\mathbf{q}, \mathbf{d}^+)$ for training in the range of $\{5\%, 10\%, 20\%, 40\%\}$, and the dimensionality of embeddings $C \in \{2, 5, 10, 20, 40\}$. We report the test results of DSE and SSI in Table 1 respectively. DSE significantly outperforms SSI when using limited word

[1]`http://www.stats.ox.ac.uk/~teh/data.html`



(a) DSE (b) SSI

Figure 1: Embeddings of the NIPS papers with the dimensionality $C = 2$, produced by DSE and SSI using 40% distinct positive links for training. The section numbers of the papers are shown as well.

Table 1: Test results (rank loss %) on NIPS data

DSE	C=2	C=5	C=10	C=20	C=50	SSI	C=2	C=5	C=10	C=20	C=50
40% train	0	0	0	0	0	40% train	13.6	7.4	6.3	9.2	14.2
20% train	0	0	0	0	0	20% train	13.3	9.5	9.4	15.5	24.9
10% train	0.01	0	0.03	0.15	1.39	10% train	16.2	11.5	14.9	22.8	29.5
5% train	0.64	0.85	3.22	8.08	17.47	5% train	20.4	19.9	25.6	31.9	35.7

features. Using just a small percentage (e.g., 10% or 20%) of training data, DSE can achieve near zero ranking loss in test. Both methods tend to overfit with little training data (e.g., $< 5\%$) and/or higher dimensionality, but the case in SSI is a lot worse. We also notice that when embedding dimension gets very small, the testing error of SSI also increase. As we found out, it is because the training errors were also high, which is a good indicator of limitation of its learning capability. In other words, SSI only exploits the linearity to induce the latent semantic embeddings and suffer from scarcity of word features, while DSE can capture the nonlinear interaction between word features to induce the latent semantic embeddings, hence alleviate the problem of feature scarcity. With more training data, both DSE and SSI perform better.

To provide anecdotal evidence that DSE can infer semantic relationships, we use each single word as input, and find K-nearest-neighbors of the word in terms of latent embeddings output by the first layer of DSE. Table 2 shows some examples, where we select 40 words such as “images”, “neuron”, “classification”, “propagation” and “maximum” for illustration. Using DSE, we can identify words that are similar in semantics or they are part of a phrase. For example, when “propagation” is taken as input, the probable words associated to it are “back”, “class”, “distributed”, “gradient” and “classifier”. They are indeed relevant in neural network literature. We may also automatically identify synthetic synonyms. For example, given the word “neuron”, we have “neuron” as the most relevant word. Other examples include “class : classes”, “representation : representations”, “images : image”, “node : nodes”, “cells : cell” and so on.

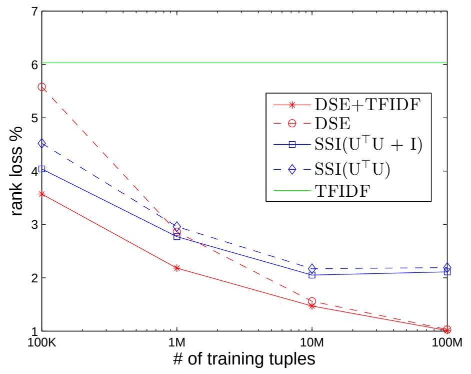


Figure 2: Test results using top 1000 most frequent words.

4.3 Wikipedia Web Pages

We then apply DSE to document ranking for a large-scale Wikipedia dataset, which consists of $1,828,645$ English Wikipedia web pages with $24,667,286$ links between them. We deem two documents with a link are relevant, and otherwise they are irrelevant. We randomly generate 70/30 splits of the links. The models are trained and validated on the 70% data, and test results on the remaining 30% data are reported. Figure 2 shows the test results using top 1,000 most frequent words, where the DNN architecture is empirically tuned as 1000-500-500-1000-100, and the normalized TFIDF features are used as input in the first layer of DNN. We report test results using different number of randomly generated training tuples. We compare DSE with SSI, and use TFIDF scheme as baseline. When using a small number of

Table 2: Interpretion of semantic relationships between words on NIPS data. We use each word as input, then find 5-nearest neighbors for each word on the semantic embedding output by the first layer of DSE.

WORD	5 Nearest Neighbors				
propagation	back	class	distributed	gradient	classifier
recognition	classification	word	classifier	mixture	speech
neuron	neurons	synaptic	firing	continuous	frequency
class	classes	classifier	finite	multi	analog
matrix	mixture	em	class	basis	vectors
visual	image	cortex	frequency	spatial	synaptic
feature	classifier	features	cortex	frequency	measure
cells	cell	firing	spatial	frequency	circuit
field	receptive	spatial	variable	images	cells
response	stimulus	frequency	orientation	analog	phase
representation	representations	firing	analog	images	statistical
images	image	orientation	features	position	visual
generalization	classifier	receptive	classification	dimension	vectors
speech	signals	multi	markov	word	frequency
states	markov	likelihood	circuit	orientation	dynamics
gradient	vectors	propagation	dimension	class	simulations
activity	stimulus	synaptic	firing	frequency	signals
variables	variable	regression	likelihood	sample	bayesian
distance	vectors	position	bayesian	specific	regression
connections	synaptic	continuous	firing	net	recurrent
synaptic	firing	approximation	sample	frequency	connections
node	nodes	spatial	bayesian	net	tree
maximum	likelihood	probabilities	mixture	classification	regression
components	component	mixture	vectors	density	continuous
outputs	vol	multi	continuous	orientation	receptive
back	propagation	continuous	classifier	circuit	multi
convergence	regression	vectors	em	long	equations
analog	circuit	class	equations	frequency	dimension
properties	stimulus	frequency	type	cells	firing
likelihood	mixture	probabilities	bayesian	log	frequency
component	components	vectors	orientation	representation	mixture
procedure	synaptic	firing	generalization	threshold	dynamics
adaptive	mixture	search	cells	likelihood	receptive
mixture	likelihood	em	variance	gaussian	multi
action	firing	search	position	scale	synaptic
circuit	analog	cells	type	firing	feedback
stimulus	frequency	orientation	firing	filter	response
bayesian	likelihood	probabilities	prior	mixture	spatial
probabilities	likelihood	bayesian	markov	mixture	type
simulations	simulation	firing	net	synaptic	recurrent

training samples (e.g., 100K tuples), DSE does not perform well, and tends to overfit. However, with sufficient training tuples, DSE consistently outperforms SSI. A linear combination of DSE output and the orginal TFIDF yields even larger performance gain. Both supervised learning methods (DSE and SSI) perform significantly better than the baseline TFIDF scheme, which does not use any label information.

We also investigate how the number of word features in the form of TFIDF affect the performance of DSE. Table 3 shows not only the linear embedding model SSI, but also a nonlinear ranking model PSI [2]. In SSI and PSI, we use the same embedding dimension of 100. Note PSI has two sets of embeddings for documents in the 3-way dotproduct (see the paper for details). We can see that that DSE outperforms SSI and PSI by a large margin when using top 500, 1000, 1500 features. The rank loss decreases as the number of input features increase.

We also observe the convergence of DSE is slow using a large number of word features (e.g., top 5,000 words) in the form of TFIDF. Instead, we exploit binary features of word-document occurrence which works well in practice in terms of convergence time, compared to the TFIDF features. With 100M tuples for training, the ranking loss of DSE using top 5,000 word features is 0.56%, while SSI yields a value of 0.76%, and TFIDF scheme has a ranking loss 2.96%. We found the performance of DSE using top 5,000 word features is significantly better than TFIDF scheme using even all the 2.5 million words which has a ranking loss 0.84%. This has practical implication that DSE may use a much reduced size of word vocabulary for indexing, while preserving the

Table 3: Test results (rank loss) on Wikipedia data using different number of top frequent words

# of words	500	1000	1500
DSE %	2.10	1.03	0.92
SSI %	3.79	2.19	1.50
PSI %	2.35	1.50	1.20

accuracy of document retrieval.

5. CONCLUSION

In this paper, we consider using deep architecture with RBMs to model the semantics in text document. We proposed Deep Semantic Embedding (DSE), which achieves nonlinear embeddings in latent semantic space for word inputs by respecting similarity of query-document pairs. DSE is trained in supervised learning on tuples with relevant and irrelevant document pairs given a query, and capable to preserving discriminative features for ranking documents. Test results on ranking scientific articles and Wikipedia web pages show the effectiveness of DSE. For future work, we would like to explore rectified linear units [17] and dropout [10] methods to speed up convergence of our model and alleviate co-adaptation of word features.

6. REFERENCES

[1] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, O. Chapelle, and K. Weinberger. Supervised semantic indexing. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 187–196. ACM, 2009.

[2] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, C. Cortes, and M. Mohri. Polynomial semantic indexing. In *NIPS*, pages 64–72, 2009.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Advances in neural information processing systems*, 1:601–608, 2002.

[4] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML 2005*, pages 89–96, New York, NY, USA, 2005. ACM Press New York, NY, USA.

[5] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face veriïňĄcation. In *CVPR*, 2005.

[6] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[7] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[8] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[9] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[10] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[11] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[12] H. Larochelle and S. Lauly. A neural autoregressive topic model. In *NIPS*, pages 2717–2725, 2012.

[13] N. Le Roux and Y. Bengio. Representational power of restricted boltzmann machines and deep belief networks. *Neural Computation*, 20(6):1631–1649, 2008.

[14] Z. Lu and H. Li. A deep architecture for matching short texts. 2013.

[15] M. R. Min, D. A. Stanley, Z. Yuan, A. J. Bonner, and Z. Zhang. A deep non-linear feature mapping for large-margin knn classification. In W. W. 0010, H. Kargupta, S. Ranka, P. S. Yu, and X. Wu, editors, *ICDM*, pages 357–366. IEEE Computer Society, 2009.

[16] M. R. Min, L. van der Maaten, Z. Yuan, A. J. Bonner, and Z. Zhang. Deep supervised t-distributed embedding. In J. FÃijrnkranz and T. Joachims, editors, *ICML*, pages 791–798. Omnipress, 2010.

[17] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[18] R. Salakhutdinov and G. E. Hinton. Replicated softmax: an undirected topic model. In *NIPS*, volume 22, pages 1607–1614, 2009.

[19] G. Salton and M. J. McGill. Introduction to modern information retrieval. 1983.

[20] A. J. Smola. *Advances in large margin classifiers.* MIT press, 2000.

[21] Y. Song, H. Wang, and X. He. Adapting deep ranknet for personalized search. In *WSDM*, 2014.

[22] N. Srivastava, R. R. Salakhutdinov, and G. E. Hinton. Modeling documents with deep boltzmann machines. In *UAI*, 2013.

[23] M. Welling, M. Rosen-Zvi, and G. E. Hinton. Exponential family harmoniums with an application to information retrieval. In *Nips*, volume 17, pages 1481–1488, 2004.