Modeling Interestingness with Deep Neural Networks

Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, Li Deng
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
{jfgao,ppantel,mgamon,xiaohe,deng}@microsoft.com

Abstract

This paper presents a deep semantic similarity model (DSSM), a special type of deep neural networks designed for text analysis, for recommending *target* documents to be of interest to a user based on a *source* document that she is reading. We observe, identify, and detect naturally occurring signals of *interestingness* in click transitions on the Web between source and target documents, which we collect from commercial Web browser logs. The DSSM is trained on millions of Web transitions, and maps source-target document pairs to feature vectors in a latent space in such a way that the distance between source documents and their corresponding interesting targets in that space is minimized. The effectiveness of the DSSM is demonstrated using two *interestingness* tasks: automatic highlighting and contextual entity search. The results on large-scale, real-world datasets show that the semantics of documents are important for modeling interestingness and that the DSSM leads to significant quality improvement on both tasks, outperforming not only the classic document models that do not use semantics but also state-of-the-art topic models.

1 Introduction

Tasks of predicting what interests a user based on the document she is reading are fundamental to many online recommendation systems. A recent survey is due to Ricci et al. (2011). In this paper, we exploit the use of a deep semantic model for two such *interestingness* tasks in which document semantics play a crucial role: automatic highlighting and contextual entity search.

Automatic Highlighting. In this task we want a recommendation system to automatically discover the entities (e.g., a person, location, organization etc.) that interest a user when reading a document and to highlight the corresponding text spans, referred to as *keywords* afterwards. We show in this study that document semantics are among the most important factors that influence what is perceived as interesting to the user. For example, we observe in Web browsing logs that when a user reads an article about a movie, she is more likely to browse to an article about an actor or character than to another movie or the director.

Contextual entity search. After identifying the keywords that represent the entities of interest to the user, we also want the system to recommend new, interesting documents by searching the Web for supplementary information about these entities. The task is challenging because the same keywords often refer to different entities, and interesting supplementary information to the highlighted entity is highly sensitive to the semantic context. For example, "Paul Simon" can refer to many people, such as the singer and the senator. Consider an article about the music of Paul Simon and another about his life. Related content about his upcoming concert tour is much more interesting in the first context, while an article about his family is more interesting in the second.

At the heart of these two tasks is the notion of interestingness. In this paper, we model and make use of this notion of interestingness with a deep semantic similarity model (DSSM). The model, extending from the deep neural networks shown recently to be highly effective for speech recognition (Hinton et al., 2012; Deng et al., 2013) and computer vision (Krizhevsky et al., 2012; Markoff, 2014), is *semantic* because it maps documents to feature vectors in a latent semantic space, also known as semantic representations. The model is *deep* because it employs a neural network with several hidden layers including a special convolutional-pooling structure to identify keywords and extract hidden semantic features at different levels of abstractions, layer by layer. The semantic representation is computed through a deep neural network after its training by back-propagation with respect to an objective tailored

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2–13,
October 25-29, 2014, Doha, Qatar.

to the respective interestingness tasks. We obtain naturally occurring "interest" signals by observing Web browser transitions, from a source document to a target document, in Web usage logs of a commercial browser. Our training data is sampled from these transitions.

The use of the DSSM to model interestingness is motivated by the recent success of applying related deep neural networks to computer vision (Krizhevshy et al. 2012; Markoff, 2014), speech recognition (Hinton et al. 2012), text processing (Collobert et al. 2011), and Web search (Huang et al. 2013). Among them, (Huang et al. 2013) is most relevant to our work. They also use a deep neural network to map documents to feature vectors in a latent semantic space. However, their model is designed to represent the *relevance* between queries and documents, which differs from the notion of interestingness between documents studied in this paper. It is often the case that a user is interested in a document because it provides supplementary information about the entities or concepts she encounters when reading another document although the overall contents of the second documents is not highly relevant. For example, a user may be interested in knowing more about the history of University of Washington after reading the news about President Obama's visit to Seattle. To better model interestingness, we extend the model of Huang et al. (2013) in two significant aspects. First, while Huang et al. treat a document as a bag of words for semantic mapping, the DSSM treats a document as a sequence of words and tries to discover prominent keywords. These keywords represent the entities or concepts that might interest users, via the convolutional and max-pooling layers which are related to the deep models used for computer vision (Krizhevsky et al., 2013) and speech recognition (Deng et al., 2013a) but are not used in Huang et al.'s model. The DSSM then forms the high-level semantic representation of the whole document based on these keywords. Second, instead of directly computing the document relevance score using cosine similarity in the learned semantic space, as in Huang et al. (2013), we feed the features derived from the semantic representations of documents to a ranker which is trained in a supervised manner. As a result, a document that is not highly relevant to another document a user is reading (i.e., the distance between their derived feature vectors is big) may still have a high score of interestingness because the former provides useful information about an entity mentioned in the latter. Such information and entity are encoded, respectively, by (some subsets of) the semantic features in their corresponding documents. In Sections 4 and 5, we empirically demonstrate that the aforementioned two extensions lead to significant quality improvements for the two interestingness tasks presented in this paper.

Before giving a formal description of the DSSM in Section 3, we formally define the *interestingness* function, and then introduce our data set of naturally occurring interest signals.

2 The Notion of Interestingness

Let D be the set of all documents. Following Gamon et al. (2013), we formally define the *interestingness* modeling task as learning the mapping function:

$$\sigma: D \times D \rightarrow \mathbb{R}^{+}$$

where the function $\sigma(s,t)$ is the quantified degree of interest that the user has in the target document $t \in D$ after or while reading the source document $s \in D$.

Our notion of a document is meant in its most general form as a string of raw unstructured text. That is, the interestingness function should not rely on any document structure such as title tags, hyperlinks, etc., or Web interaction data. In our tasks, documents can be formed either from the plain text of a webpage or as a text span in that plain text, as will be discussed in Sections 4 and 5.

2.1 Data

We can observe many naturally occurring manifestations of interestingness on the Web. For example, on Twitter, users follow shared links embedded in tweets. Arguably the most frequent signal, however, occurs in Web browsing events where users click from one webpage to another via hyperlinks. When a user clicks on a hyperlink, it is reasonable to assume that she is interested in learning more about the anchor, modulo cases of erroneous clicks. Aggregate clicks can therefore serve as a proxy for interestingness. That is, for a given source document, target documents that attract the most clicks are more interesting than documents that attract fewer clicks[1].

[1] We stress here that, although the click signal is available to form a dataset and a gold standard ranker (to be described in Section 4), our task is to model interestingness between unstructured documents, i.e., without access to any document structure or Web interaction data. Thus, in our experiments,

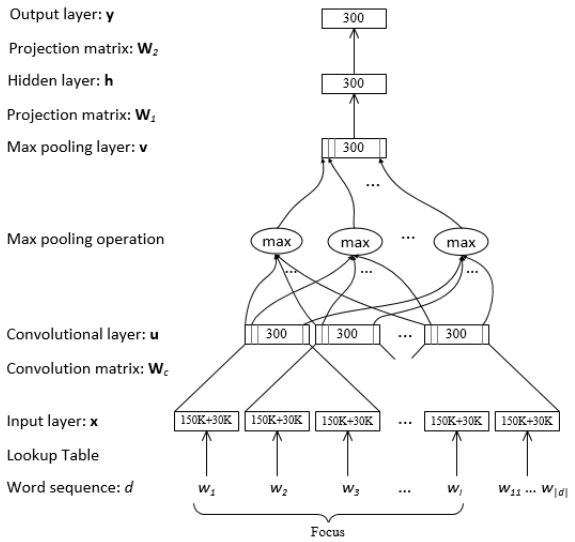

Figure 1: Illustration of the network architecture and information flow of the DSSM

We collect a large dataset of user browsing events from a commercial Web browser. Specifically, we sample 18 million occurrences of a user click from one Wikipedia page to another during a one year period. We restrict our browsing events to Wikipedia since its pages tend to contain many anchors (79 on average, where on average 42 have a unique target URL). Thus, they attract enough traffic for us to obtain robust browsing transition data[2]. We group together all transitions originating from the same page and randomly hold out 20% of the transitions for our evaluation data (**EVAL**), 20% for training the DSSM described in Section 3.2 (**TRAIN_1**), and the remaining 60% for training our task specific rankers described in Section 3.3 (**TRAIN_2**). In our experiments, we used different settings for the two interestingness tasks. Thus, we postpone the detailed description of these datasets and other task-specific datasets to Sections 4 and 5.

3 A Deep Semantic Similarity Model (DSSM)

This section presents the architecture of the DSSM, describes the parameter estimation, and the way the DSSM is used in our tasks.

3.1 Network Architecture

The heart of the DSSM is a deep neural network with convolutional structure, as shown in Figure 1. In what follows, we use lower-case bold letters, such as $\mathbf{x}$, to denote column vectors, $x(i)$ to denote the i^{th} element of $\mathbf{x}$, and upper-case letters, such as $\mathbf{W}$, to denote matrices.

Input Layer x. It takes two steps to convert a document d, which is a sequence of words, into a vector representation $\mathbf{x}$ for the input layer of the network: (1) convert each word in d to a word vector, and (2) build $\mathbf{x}$ by concatenating these word vectors. To convert a word w into a word vector, we first represent w by a one-hot vector using a vocabulary that contains N high frequent words ($N = 150$K in this study). Then, following Huang et al. (2013), we map w to a separate tri-letter vector. Consider the word "#dog#", where # is a word boundary symbol. The nonzero elements in its tri-letter vector are "#do", "dog", and "og#". We then form the word vector of w by concatenating its one-hot vector and its tri-letter vector. It is worth noting that the tri-letter vector complements the one-hot vector representation in two aspects. First, different OOV (out of vocabulary) words can be represented by tri-letter vectors with few collisions. Second, spelling variations of the same word can be mapped to the points that are close to each other in the tri-letter space. Although the number of unique English words on the Web is extremely large, the total number of distinct tri-letters in English is limited (restricted to the most frequent 30K in this study). As a result, incorporating tri-letter vectors substantially improves the representation power of word vectors while keeping their size small.

To form our input layer $\mathbf{x}$ using word vectors, we first identify a text span with a high degree of relevance, called *focus*, in d using task-specific heuristics (see Sections 4 and 5 respectively). Second, we form $\mathbf{x}$ by concatenating each word vector in the focus and a vector that is the summation of all other word vectors, as shown in Figure 1. Since the length of the focus is much smaller than that of its document, $\mathbf{x}$ is able to capture the contextual information (for the words in the focus)

we remove all structural information (e.g., hyperlinks and XML tags) in our documents, except that in the highlighting experiments (Section 4) we use anchor texts to simulate the candidate keywords to be highlighted. We then convert each Web document into plain text, which is white-space tokenized and lowercased. Numbers are retained and no stemming is performed.

[2] We utilize the May 3, 2013 English Wikipedia dump consisting of roughly 4.1 million articles from http://dumps.wikimedia.org.

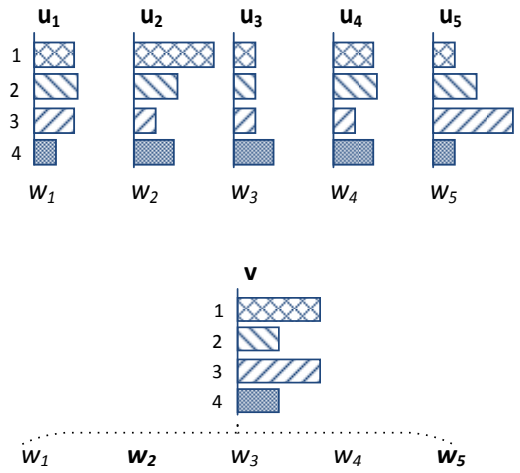

Figure 2: Toy example of (upper) a 5-word document and its local feature vectors extracted using a convolutional layer, and (bottom) the global feature vector of the document generated after max-pooling.

useful to the corresponding tasks, with a manageable vector size.

Convolutional Layer u. A convolutional layer extracts local features around each word w_i in a word sequence of length I as follows. We first generate a contextual vector $\mathbf{c}_i$ by concatenating the word vectors of w_i and its surrounding words defined by a window (the window size is set to 3 in this paper). Then, we generate for each word a local feature vector $\mathbf{u}_i$ using a tanh activation function and a linear projection matrix $\mathbf{W}_c$, which is the same across all windows i in the word sequence, as:

$$\mathbf{u}_i = \tanh\left(\mathbf{W}_c^{\mathrm{T}}\mathbf{c}_i\right), \text{where } i = 1 \dots I \quad (1)$$

Max-pooling Layer v. The size of the output $\mathbf{u}$ depends on the number of words in the word sequence. Local feature vectors have to be combined to obtain a global feature vector, with a fixed size independent of the document length, in order to apply subsequent standard affine layers. We design $\mathbf{v}$ by adopting the max operation over each "time" i of the sequence of vectors computed by (1), which forces the network to retain only the most useful, partially invariant local features produced by the convolutional layer:

$$v(j) = \max_{i=1,\dots,I}\{\mathrm{u}_i(j)\} \quad (2)$$

where the max operation is performed for each dimension of $\mathbf{u}$ across $i = 1, \dots, I$ respectively.

That convolutional and max-pooling layers are able to discover prominent keywords of a document can be demonstrated using the procedure in Figure 2 using a toy example. First, the convolutional layer of (1) generates for each word in a 5-word document a 4-dimensional local feature vector, which represents a distribution of four *topics*. For example, the most prominent topic of w_2 within its three word context window is the first topic, denoted by $u_2(1)$, and the most prominent topic of w_5 is $u_5(3)$. Second, we use max-pooling of (2) to form a global feature vector, which represents the topic distribution of the whole document. We see that $v(1)$ and $v(3)$ are two prominent topics. Then, for each prominent topic, we trace back to the local feature vector that survives max-pooling:

$$v(1) = \max_{i=1,\dots,5}\{u_i(1)\} = u_2(1)$$

$$v(3) = \max_{i=1,\dots,5}\{u_i(3)\} = u_5(3).$$

Finally, we label the corresponding words of these local feature vectors, w_2 and w_5, as keywords of the document.

Figure 3 presents a sample of document snippets and their keywords detected by the DSSM according to the procedure elaborated in Figure 2. It is interesting to see that many names are identified as keywords although the DSSM is not designed explicitly for named entity recognition.

Fully-Connected Layers h and y. The fixed sized global feature vector $\mathbf{v}$ of (2) is then fed to several standard affine network layers, which are stacked and interleaved with nonlinear activation functions, to extract highly non-linear features $\mathbf{y}$ at the output layer. In our model, shown in Figure 1, we have:

$$\mathbf{h} = \tanh(\mathbf{W}_1^{\mathrm{T}}\mathbf{v}) \quad (3)$$

$$\mathbf{y} = \tanh(\mathbf{W}_2^{\mathrm{T}}\mathbf{h}) \quad (4)$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are learned linear projection matrices.

3.2 Training the DSSM

To optimize the parameters of the DSSM of Figure 1, i.e., $\boldsymbol{\theta} = \{\mathbf{W}_c, \mathbf{W}_1, \mathbf{W}_2\}$, we use a pair-wise rank loss as objective (Yih et al. 2011). Consider a source document s and two candidate target documents t_1 and t_2, where t_1 is more interesting than t_2 to a user when reading s. We construct two pairs of documents (s, t_1) and (s, t_2), where the former is preferred and should have a higher

… the **comedy festival** formerly known as the us **comedy arts** festival is a comedy festival held each year in **las vegas nevada** from its 1985 inception to 2008 . it was held annually at the **wheeler opera house** and other venues in **aspen colorado** . the primary sponsor of the festival was hbo with co-sponsorship by caesars palace . the primary venue tbs **geico insurance** twix candy bars and **smirnoff vodka hbo** exited the festival business in 2007 and tbs became the primary sponsor the festival includes standup comedy performances appearances by the casts of television shows…

… **bad samaritans** is an american **comedy** series produced by **walt becker kelly** hayes and **ross putman** . it premiered on **netflix** on march 31 2013 cast and characters . the show focuses on a community service parole group and their parole officer **brian kubach** as **jake gibson** an aspiring professional **starcraft** player who gets sentenced to 2000 hours of community service for starting a **forest fire** during his **breakup** with **drew** prior to community service he had no real ambition in life other than to be a professional gamer and become wealthy overnight like **mark zuckerberg** as in life his goal during …

Figure 3: A sample of document snippets and the keywords (in **bold**) detected by the DSSM.

interestingness score. Let Δ be the difference of their interestingness scores: $\Delta = \sigma(s, t_1) - \sigma(s, t_2)$, where σ is the interestingness score, computed as the cosine similarity:

$$\sigma(s,t) \equiv \text{sim}_{\boldsymbol{\theta}}(s,t) = \frac{\mathbf{y}_s^{\mathrm{T}}\mathbf{y}_t}{\|\mathbf{y}_s\|\|\mathbf{y}_t\|} \qquad (5)$$

where $\mathbf{y}_s$ and $\mathbf{y}_t$ are the feature vectors of s and t, respectively, which are generated using the DSSM, parameterized by $\boldsymbol{\theta}$. Intuitively, we want to learn $\boldsymbol{\theta}$ to maximize Δ. That is, the DSSM is learned to represent documents as points in a hidden interestingness space, where the similarity between a document and its interesting documents is maximized.

We use the following logistic loss over Δ, which can be shown to upper bound the pairwise accuracy:

$$\mathcal{L}(\Delta; \boldsymbol{\theta}) = \log(1 + \exp(-\gamma\Delta)) \qquad (6)$$

The loss function in (6) has a shape similar to the hinge loss used in SVMs. Because of the use of the cosine similarity function, we add a scaling factor γ that magnifies Δ from [-2, 2] to a larger range. Empirically, the value of γ makes no difference as long as it is large enough. In the experiments, we set $\gamma = 10$. Because the loss function is differentiable, optimizing the model parameters can be done using gradient-based methods. Due to space limitations, we omit the derivation of the gradient of the loss function, for which readers are referred to related derivations (e.g., Collobert et al. 2011; Huang et al. 2013; Shen et al. 2014).

In our experiments we trained DSSMs using mini-batch Stochastic Gradient Descent. Each mini-batch consists of 256 source-target document pairs. For each source document s, we randomly select from that batch four target documents which are not paired with s as negative training samples[3]. The DSSM trainer is implemented using a GPU-accelerated linear algebra library, which is developed on CUDA 5.5. Given the training set (**TRAIN_1** in Section 2), it takes approximately 30 hours to train a DSSM as shown in Figure 1, on a Xeon E5-2670 2.60GHz machine with one Tesla K20 GPU card.

In principle, the loss function of (6) can be further regularized (e.g. by adding a term of $L2$ norm) to deal with overfitting. However, we did not find a clear empirical advantage over the simpler early stop approach in a pilot study, hence we adopted the latter in the experiments in this paper. Our approach adjusts the learning rate η during the course of model training. Starting with $\eta = 1.0$, after each epoch (a pass over the entire training data), the learning rate is adjusted as $\eta = 0.5 \times \eta$ if the loss on validation data (held-out from **TRAIN_1**) is not reduced. The training stops if either η is smaller than a preset threshold (0.0001) or the loss on training data can no longer be reduced significantly. In our experiments, the DSSM training typically converges within 20 epochs.

3.3 Using the DSSM

We experiment with two ways of using the DSSM for the two interestingness tasks. First, we use the DSSM as a feature generator. The output layer of the DSSM can be seen as a set of semantic features, which can be incorporated in a boosted tree

[3] In our experiments, we observed better results by sampling more negative training examples (e.g., up to 100) although this makes the training much slower. An alternative approach is to approximate the partition function using Noise Contrastive Estimation (Gutmann and Hyvarinen 2010). We leave it to future work.

	#	Models	HEAD			TORSO			TAIL		
			@1	@5	@10	@1	@5	@10	@1	@5	@10
src only	1	**RAND**	0.041	0.062	0.081	0.036	0.076	0.109	0.062	0.195	0.258
	2	**1stK**	0.010	0.177	0.243	0.072	0.171	0.240	0.091	0.274	0.348
	3	**LastK**	0.170	0.022	0.027	0.022	0.044	0.062	0.058	0.166	0.219
	4	**NSF**	0.215	0.253	0.295	0.139	0.229	0.282	0.109	0.293	0.365
	5	**NSF+WCAT**	0.438	0.424	0.463	0.194	0.290	0.346	0.118	0.317	0.386
	6	**NSF+JTT**	0.220	0.302	0.343	0.141	0.241	0.295	0.111	0.300	0.369
	7	**NSF+DSSM_BOW**	0.312	0.351	0.391	0.162	0.258	0.313	0.110	0.299	0.372
	8	**NSF+DSSM**	0.362	0.386	0.421	0.178	0.275	0.330	0.116	0.312	0.382
src+tar	9	**NSF+WCAT**	0.505	0.475	0.501	0.224	0.304	0.356	0.129	0.324	0.391
	10	**NSF+JTT**	0.345	0.380	0.418	0.183	0.280	0.332	0.131	0.321	0.390
	11	**NSF+DSSM_BOW**	0.416	0.393	0.428	0.197	0.274	0.325	0.123	0.311	0.380
	12	**NSF+DSSM**	**0.554**	**0.524**	**0.547**	**0.241**	**0.317**	**0.367**	**0.135**	**0.329**	**0.398**

Table 1: Highlighting task performance (NDCG @ K) of interest models over HEAD, TORSO and TAIL test sets. Bold indicates statistical significance over all non-shaded results using *t*-test ($p = 0.05$).

based ranker (Friedman 1999) trained discriminatively on the task-specific data. Given a source-target document pair (s, t), the DSSM generates 600 features (300 from the output layers $\mathbf{y}_s$ and $\mathbf{y}_t$ for each s and t, respectively).

Second, we use the DSSM as a direct implementation of the interestingness function σ. Recall from Section 3.2 that in model training, we measure the interestingness score for a document pair using the cosine similarity between their corresponding feature vectors ($\mathbf{y}_s$ and $\mathbf{y}_t$). Similarly at runtime, we define $\sigma = \text{sim}_{\boldsymbol{\theta}}(s, t)$ as (5).

4 Experiments on Highlighting

Recall from Section 1 that in this task, a system must select k most interesting keywords in a document that a user is reading. To evaluate our models using the click transition data described in Section 2, we simulate the task as follows. We use the set of anchors in a source document s to simulate the set of candidate keywords that may be of interest to the user while reading s, and treat the text of a document that is linked by an anchor in s as a target document t. As shown in Figure 1, to apply DSSM to a specific task, we need to define the focus in source and target documents. In this task, the *focus* in *s* is defined as the anchor text, and the *focus* in *t* is defined as the first 10 tokens in *t*.

We evaluate the performance of a highlighting system against a gold standard interestingness function σ' which scores the interestingness of an anchor as the number of user clicks on t from the anchor in s in our data. We consider the ideal selection to then consist of the k most interesting anchors according to σ'. A natural metric for this task is Normalized Discounted Cumulative Gain (NDCG) (Jarvelin and Kekalainen 2000).

We evaluate our models on the **EVAL** dataset described in Section 2. We utilize the transition distributions in **EVAL** to create three other test sets, following the stratified sampling methodology commonly employed in the IR community, for the frequently, less frequently, and rarely viewed source pages, referred to as **HEAD**, **TORSO**, and **TAIL**, respectively. We obtain these sets by first sorting the unique source documents according to their frequency of occurrence in **EVAL**. We then partition the set so that **HEAD** corresponds to all transitions from the source pages at the top of the list that account for 20% of the transitions in **EVAL**; **TAIL** corresponds to the transitions at the bottom also accounting for 20% of the transitions in **EVAL**; and **TORSO** corresponds to the remaining transitions.

4.1 Main Results

Table 1 summarizes the results of various models over the three test sets using NDCG at truncation levels 1, 5, and 10.

Rows 1 to 3 are simple heuristic baselines. **RAND** selects k random anchors, **1stK** selects the first k anchors and **LastK** the last k anchors.

The other models in Table 1 are boosted tree based rankers trained on **TRAIN_2** described in Section 2. They vary only in their features. The ranker in Row 4 uses Non-Semantic Features (**NSF**) only. These features are derived from the

source document *s* and from user session information in the browser log. The document features include: position of the anchor in the document, frequency of the anchor, and anchor density in the paragraph.

The rankers in Rows 5 to 12 use the NSF and the semantic features computed from source and target documents of a browsing transition. We compare semantic features derived from three different sources. The first feature source comes from our DSSMs (**DSSM** and **DSSM_BOW**) using the output layers as feature generators as described in Section 3.3. **DSSM** is the model described in Section 3 and **DSSM_BOW** is the model proposed by Huang et al. (2013) where documents are view as bag of words (BOW) and the convolutional and max-pooling layers are not used. The two other sources of semantic features are used as a point of comparison to the DSSM. One is a generative semantic model (Joint Transition Topic model, or **JTT**) (Gamon et al. 2013). **JTT** is an LDA-style model (Blei et al. 2003) that is trained jointly on source and target documents linked by browsing transitions. **JTT** generates a total of 150 features from its latent variables, 50 each for the source topic model, the target topic model and the transition model. The other semantic model of contrast is a manually defined one, which we use to assess the effectiveness of automatically learned models against human modelers. To this effect, we use the page categories that editors assign in Wikipedia as semantic features (**WCAT**). These features number in the multiple thousands. Using features such as **WCAT** is not a viable solution in general since Wikipedia categories are not available for all documents. As such, we use it solely as a point of comparison against **DSSM** and **JTT**.

We also distinguish between two types of learned rankers: those which draw their features only from the *source* (**src only**) document and those that draw their features from both the *source* and *target* (**src+tar**) documents. Although our task setting allows access to the content of both source and target documents, there are practical scenarios where a system should predict what interests the user without looking at the target document because the extra step of identifying a suitable target document for each candidate concept or entity of interest is computationally expensive.

4.2 Analysis of Results

As shown in Table 1, **NSF+DSSM**, which incorporates our DSSM, is the overall best performing system across test sets. The task is hard as evidenced by the weak baseline scores. One reason is the large average number of candidates per page. On **HEAD**, we found an average of 170 anchors (of which 95 point to a unique target URL). For **TORSO** and **TAIL**, we found the average number of anchors to be 94 (52 unique targets) and 41 (19 unique targets), respectively.

Clearly, the semantics of the documents form important signals for this task: **WCAT**, **JTT**, **DSSM_BOW**, and **DSSM** all significantly boost the performance over NSF alone. There are two interesting comparisons to consider: (a) manual semantics vs. learned semantics; and (b) deep semantic models vs. generative topic models. On (a), we observe somewhat surprisingly that the learned DSSM produces features that outperform the thousands of features coming from manually (editor) assigned Wikipedia category features (**WCAT**), in all but the TAIL where the two perform statistically the same. In contrast, features from the generative model (**JTT**) perform worse than **WCAT** across the board except on **TAIL** where **JTT** and **WCAT** are statistically tied. On (b), we observe that DSSM outperforms a state-of-the-art generative model (**JTT**) on **HEAD** and **TORSO**. On **TAIL**, they are statistically indistinguishable.

We turn now to inspecting the scenario where features are only drawn from the source document (Rows 1-8 in Table 1). Again we observe that semantic features significantly boost the performance against **NSF** alone, however they significantly deteriorate when compared to using features from both source and target documents. In this scenario, the manual semantics from **WCAT** outperform all other models, but with a diminishing effect as we move from **HEAD** through **TORSO** to **TAIL**. **DSSM** is the best performing learned semantic model.

Finally, we present the results to justify the two modifications we made to extend the model of Huang et al. (2013) to the DSSM, as described in Section 1. First, we see in Table 1 that **DSSM_BOW**, which has the same network structure of Huang et al.'s model, is much weaker than **DSSM**, demonstrating the benefits of using convolutional and max-pooling layers to extract semantic features for the highlighting task. Second, we conduct several experiments by using the cosine scores between the output layers of **DSSM** for s and t as features (following the procedure in Section 3.3 for using the DSSM as a direct implementation of σ). We found that adding the cosine

#	Models	@1	@3	AUC
1	**BM25 (entity)**	0.133	0.195	0.583
2	**BM25**	0.142	0.227	0.675
3	**WTM**	0.191	0.287	0.678
4	**BLTM**	0.214	0.306	0.704
5	**DSSM**	0.259*	0.356*	0.711*
6	**DSSM_BOW**	0.223	0.322	0.699
7	**Baseline ranker**	0.283	0.360	0.723
8	**7 + DSSM(1)**	0.301#	0.385#	0.758#
9	**7 + DSSM(600)**	0.327##	0.402##	0.782##

Table 2: Contextual entity search task performance (NDCG @ K and AUC). * indicates statistical significance over all non-shaded single model results (Rows 1 to 6) using *t*-test ($p < 0.05$). # indicates statistical significance over results in Row 7. ## indicates statistical significance over results in Rows 7 and 8.

features to **NSF+DSSM** does not lead to any improvement. We also combined **NSF** with solely the cosine features from **DSSM** (i.e., without the other semantic features drawn from its output layers). But we still found no improvement over using **NSF** alone. Thus, we conclude that for this task it is much more effective to feed the features derived from **DSSM** to a supervised ranker than directly computing the interestingness score using cosine similarity in the learned semantic space, as in Huang et al. (2013).

5 Experiments on Entity Search

We construct the evaluation data set for this second task by randomly sampling a set of documents from a traffic-weighted set of Web documents. In a second step, we identify the entity names in each document using an in-house named entity recognizer. We issue each entity name as a query to a commercial search engine, and retain up to the top-100 retrieved documents as candidate *target* documents. We form for each entity a *source* document which consists of the entity text and its surrounding text defined by a 200-word window. We define the *focus* (as in Figure 1) in s as the entity text, and the *focus* in t as the first 10 tokens in t.

The final evaluation data set contains 10,000 source documents. On average, each source document is associated with 87 target documents. Finally, the source-target document pairs are labeled in terms of interestingness by paid annotators. The label is on a 5-level scale, 0 to 4, with 4 meaning the target document is the most interesting to the source document and 0 meaning the target is of no interest.

We test our models on two scenarios. The first is a ranking scenario where k interesting documents are displayed to the user. Here, we select the top-k ranked documents according to their interestingness scores. We measure the performance via NDCG at truncation levels 1 and 3. The second scenario is to display to the user all interesting results. In this scenario, we select all target documents with an interestingness score exceeding a predefined threshold. We evaluate this scenario using ROC analysis and, specifically, the area under the curve (AUC).

5.1 Main Results

The main results are summarized in Table 2. Rows 1 to 6 are **single model** results, where each model is used as a direct implementation of the interestingness function σ. Rows 7 to 9 are **ranker** results, where σ is defined as a boosted tree based ranker that incorporates different sets of features extracted from source and target documents, including the features derived from single models. As in the highlighting experiments, all the machine-learned single models, including the DSSM, are trained on **TRAIN_1**, and all the rankers are trained on **TRAIN_2**.

5.2 Analysis of Results

BM25 (Rows 1 and 2 in Table 2) is the classic document model (Robertson and Zaragoza 2009). It uses the bag-of-words document representation and the BM25 term weighting function. In our setting, we define the interestingness score of a document pair as the dot product of their BM25-weighted term vectors. To verify the importance of using contextual information, we compare two different ways of forming the term vector of a source document. The first only uses the entity text (Row 1). The second (Row 2) uses both the entity text and and its surrounding text in a 200-word window (i.e., the entire *source* document). Results show that the model using contextual information is significantly better. Therefore, all the other models in this section use both the entity texts and their surrounding text.

WTM (Row 3) is our implementation of the word translation model for IR (Berger and Lafferty 1999; Gao et al. 2010). **WTM** defines the interestingness score as:

$$\sigma(s,t) = \prod_{w_t \in t} \sum_{w_s \in s} P(w_t|w_s) P(w_s|s),$$

where $P(w_s|s)$ is the unigram probability of word w_s in s, and $P(w_t|w_s)$ is the probability of *translating* w_s into w_t, trained on source-target document pairs using EM (Brown et al. 1993). The translation-based approach allows any pair of non-identical but semantically related words to have a nonzero matching score. As a result, it significantly outperforms **BM25**.

BTLM (Row 4) follows the best performing bilingual topic model described in Gao et al. (2011), which is an extension of PLSA (Hofmann 1999). The model is trained on source-target document pairs using the EM algorithm with a constraint enforcing a source document s and its target document t to not only share the same prior topic distribution, but to also have similar fractions of words assigned to each topic. **BLTM** defines the interestingness score between s and t as:

$$\sigma(s,t) = \prod_{w_t \in t} \sum_z P(w_t|\phi_z) P(z|\theta^s).$$

The model assumes the following story of generating t from s. First, for each topic z a word distribution ϕ_z is selected from a Dirichlet prior with concentration parameter β. Second, given s, a topic distribution θ^s is drawn from a Dirichlet prior with parameter α. Finally, t is generated word by word. Each word w_t is generated by first selecting a topic z according to θ^s, and then drawing a word from ϕ_z. We see that **BLTM** models interestingness by taking into account the semantic topic distribution of the entire documents. Our results in Table 2 show that **BLTM** outperforms **WTM** by a significant margin in both NDCG and AUC.

DSSM (Row 5) outperforms all the competing single models, including the state-of-the-art topic model **BLTM**. Now, we inspect the difference between **DSSM** and **BLTM** in detail. Although both models strive to generate the semantic representation of a document, they use different modeling approaches. **BLTM** by nature is a generative model. The semantic representation in **BLTM** is a distribution of hidden semantic topics. Such a distribution is learned using Maximum Likelihood Estimation in an unsupervised manner, i.e., maximizing the log-likelihood of the source-target document pairs in the training data. On the other hand, **DSSM** represents documents as points in a hidden semantic space using a supervised learning method, i.e., paired documents are closer in that latent space than unpaired ones. We believe that the superior performance of **DSSM** is largely due to the fact that the model parameters are discriminatively trained using an objective that is tailored to the interestingness task.

In addition to the difference in training methods, **DSSM** and **BLTM** also use different model structures. **BLTM** treats a document as a bag of words (thus losing some important contextual information such as word order and inter-word dependencies), and generates semantic representations of documents using linear projection. **DSSM**, on the other hand, treats text as a sequence of words and better captures local and global context, and generates highly non-linear semantic features via a deep neural network. To further verify our analysis, we inspect the results of a variant of **DSSM**, denoted as **DSSM_BOW** (Row 6), where the convolution and max-pooling layers are removed. This model treats a document as a bag of words, just like **BLTM**. These results demonstrate that the effectiveness of **DSSM** can also be attributed to the convolutional architecture in the neural network, in addition to being deep and being discriminative.

We turn now to discussing the ranker results in Rows 7 to 9. The baseline ranker (Row 7) uses 158 features, including many counts and single model scores, such as **BM25** and **WMT**. **DSSM** (Row 5) alone is quite effective, being close in performance to the baseline ranker with non-DSSM features. Integrating the DSSM score computed in (5) as one single feature into the ranker (Row 8) leads to a significant improvement over the baseline. The best performing combination (Row 9) is obtained by incorporating the DSSM feature vectors of source and target documents (i.e., 600 features in total) in the ranker.

We thus conclude that on both tasks, automatic highlighting and contextual entity search, features drawn from the output layers of our deep semantic model result in significant gains after being added to a set of non-semantic features, and in comparison to other types of semantic models used in the past.

6 Related Work

In addition to the notion of relevance as described in Section 1, related to interestingness is also the notion of *salience* (also called *aboutness*) (Gamon et al. 2013; 2014; Parajpe 2009; Yih et al. 2006). Salience is the centrality of a term to the content of a document. Although salience and interestingness interact, the two are not the same. For example, in a news article about President Obama's visit to Seattle, Obama is salient, yet the average user would probably not be interested in learning more about Obama while reading that article.

There are many systems that identify popular content in the Web or recommend content (e.g., Bandari et al. 2012; Lerman and Hogg 2010; Szabo and Huberman 2010), which is closely related to the highlighting task. In contrast to these approaches, we strive to predict what term a user is likely to be interested in when reading content, which may or may not be the same as the most popular content that is related to the current document. It has empirically been demonstrated in Gamon et al. (2013) that popularity is in fact a rather poor predictor for interestingness. The task of contextual entity search, which is formulated as an information retrieval problem in this paper, is also related to research on entity resolution (Stefanidis et al. 2013).

Latent Semantic Analysis (Deerwester et al. 1990) is arguably the earliest semantic model designed for IR. Generative topic models widely used for IR include PLSA (Hofmann 1990) and LDA (Blei et al. 2003). Recently, these models have been extended to handle cross-lingual cases, where there are pairs of corresponding documents in different languages (e.g., Dumais et al. 1997; Gao et al. 2011; Platt et al. 2010; Yih et al. 2011).

By exploiting deep architectures, deep learning techniques are able to automatically discover from training data the hidden structures and the associated features at different levels of abstraction useful for a variety of tasks (e.g., Collobert et al. 2011; Hinton et al. 2012; Socher et al. 2012; Krizhevsky et al., 2012; Gao et al. 2014). Hinton and Salakhutdinov (2010) propose the most original approach based on an unsupervised version of the deep neural network to discover the hierarchical semantic structure embedded in queries and documents. Huang et al. (2013) significantly extends the approach so that the deep neural network can be trained on large-scale query-document pairs giving much better performance. The use of the convolutional neural network for text processing, central to our DSSM, was also described in Collobert et al. (2011) and Shen et al. (2014) but with very different applications. The DSSM described in Section 3 can be viewed as a variant of the deep neural network models used in these previous studies.

7 Conclusions

Modeling interestingness is fundamental to many online recommendation systems. We obtain naturally occurring interest signals by observing Web browsing transitions where users click from one webpage to another. We propose to model this "interestingness" with a deep semantic similarity model (DSSM), based on deep neural networks with special convolutional-pooling structure, mapping source-target document pairs to feature vectors in a latent semantic space. We train the DSSM using browsing transitions between documents. Finally, we demonstrate the effectiveness of our model on two interestingness tasks: automatic highlighting and contextual entity search. Our results on large-scale, real-world datasets show that the semantics of documents computed by the DSSM are important for modeling interestingness and that the new model leads to significant improvements on both tasks. DSSM is shown to outperform not only the classic document models that do not use (latent) semantics but also state-of-the-art topic models that do not have the deep and convolutional architecture characterizing the DSSM.

One area of future work is to extend our method to model interestingness given an entire user session, which consists of a sequence of browsing events. We believe that the prior browsing and interaction history recorded in the session provides additional signals for predicting interestingness. To capture such signals, our model needs to be extended to adequately represent time series (e.g., causal relations and consequences of actions). One potentially effective model for such a purpose is based on the architecture of recurrent neural networks (e.g., Mikolov et al. 2010; Chen and Deng, 2014), which can be incorporated into the deep semantic model proposed in this paper.

Additional Authors

Yelong Shen (Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA, email: yeshen@microsoft.com).

Acknowledgments

The authors thank Johnson Apacible, Pradeep Chilakamarri, Edward Guo, Bernhard Kohlmeier, Xiaolong Li, Kevin Powell, Xinying Song and Ye-Yi Wang for their guidance and valuable discussions. We also thank the three anonymous reviewers for their comments.

References

Bandari, R., Asur, S., and Huberman, B. A. 2012. The pulse of news in social media: forecasting popularity. In *ICWSM*.

Bengio, Y., 2009. Learning deep architectures for AI. *Fundamental Trends in Machine Learning*, 2(1):1–127.

Berger, A., and Lafferty, J. 1999. Information retrieval as statistical translation. In *SIGIR*, pp. 222-229.

Blei, D. M., Ng, A. Y., and Jordan, M. J. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3.

Broder, A., Fontoura, M., Josifovski, V., and Riedel, L. 2007. A semantic approach to contextual advertising. In *SIGIR*.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., and Mercer, R. L. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263-311.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, and Hullender, G. 2005. Learning to rank using gradient descent. In *ICML*, pp. 89-96.

Chen, J. and Deng, L. 2014. A primal-dual method for training recurrent neural networks constrained by the echo-state property. In *ICLR*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P., 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, vol. 12.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T., and Harshman, R. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6): 391-407

Deng, L., Hinton, G., and Kingsbury, B. 2013. New types of deep neural network learning for speech recognition and related applications: An overview. In *ICASSP*.

Deng, L., Abdel-Hamid, O., and Yu, D., 2013a. A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion. In *ICASSP*.

Dumais, S. T., Letsche, T. A., Littman, M. L., and Landauer, T. K. 1997. Automatic cross-linguistic information retrieval using latent semantic indexing. In *AAAI-97 Spring Symposium Series: Cross-Language Text and Speech Retrieval*.

Friedman, J. H. 1999. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29:1189-1232.

Gamon, M., Mukherjee, A., Pantel, P. 2014. Predicting interesting things in text. In *COLING*.

Gamon, M., Yano, T., Song, X., Apacible, J. and Pantel, P. 2013. Identifying salient entities in web pages. In *CIKM*.

Gao, J., He, X., and Nie, J-Y. 2010. Clickthrough-based translation models for web search: from word models to phrase models. In *CIKM*. pp. 1139-1148.

Gao, J., He, X., Yih, W-t., and Deng, L. 2014. Learning continuous phrase representations for translation modeling. In *ACL*.

Gao, J., Toutanova, K., Yih., W-T. 2011. Clickthrough-based latent semantic models for web search. In *SIGIR*. pp. 675-684.

Graves, A., Mohamed, A., and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *ICASSP*.

Gutmann, M. and Hyvarinen, A. 2010. Noise-contrastive estimation: a new estimation principle for unnormalized statistical models. In *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS2010)*.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B., 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29:82-97.

Hinton, G., and Salakhutdinov, R., 2010. Discovering binary codes for documents by learning deep generative models. *Topics in Cognitive Science*, pp. 1-18.

Hofmann, T. 1999. Probabilistic latent semantic indexing. In *SIGIR*. pp. 50-57.

Huang, P., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*.

Jarvelin, K. and Kekalainen, J. 2000. IR evaluation methods for retrieving highly relevant documents. In *SIGIR*. pp. 41-48.

Krizhevsky, A., Sutskever, I. and Hinton, G. 2012. ImageNet classification with deep convolutional neural networks. In *NIPS*.

Lerman, K., and Hogg, T. 2010. Using a model of social dynamics to predict popularity of news. In *WWW*. pp. 621-630.

Markoff, J. 2014. Computer eyesight gets a lot more accurate. In New York Times.

Mikolov, T.. Karafiat, M., Burget, L., Cernocky, J., and Khudanpur, S. 2010. Recurrent neural network based language model. In *INTERSPEECH*. pp. 1045-1048.

Paranjpe, D. 2009. Learning document aboutness from implicit user feedback and document structure. In *CIKM*.

Platt, J., Toutanova, K., and Yih, W. 2010. Translingual document representations from discriminative projections. In *EMNLP*. pp. 251-261.

Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B. (eds) 2011. *Recommender System Handbook*, Springer.

Robertson, S., and Zaragoza, H. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333-389.

Shen, Y., He, X., Gao. J., Deng, L., and Mesnil, G. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *CIKM*.

Socher, R., Huval, B., Manning, C., Ng, A., 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*.

Stefanidis, K., Efthymiou, V., Herschel, M., and Christophides, V. 2013. Entity resolution in the web of data. *CIKM'13 Tutorial*.

Szabo, G., and Huberman, B. A. 2010. Predicting the popularity of online content. *Communications of the ACM*, 53(8).

Wu, Q., Burges, C.J.C., Svore, K., and Gao, J. 2009. Adapting boosting for information retrieval measures. *Journal of Information Retrieval*, 13(3):254-270.

Yih, W., Goodman, J., and Carvalho, V. R. 2006. Finding advertising keywords on web pages. In *WWW*.

Yih, W., Toutanova, K., Platt, J., and Meek, C. 2011. Learning discriminative projections for text similarity measures. In *CoNLL*.