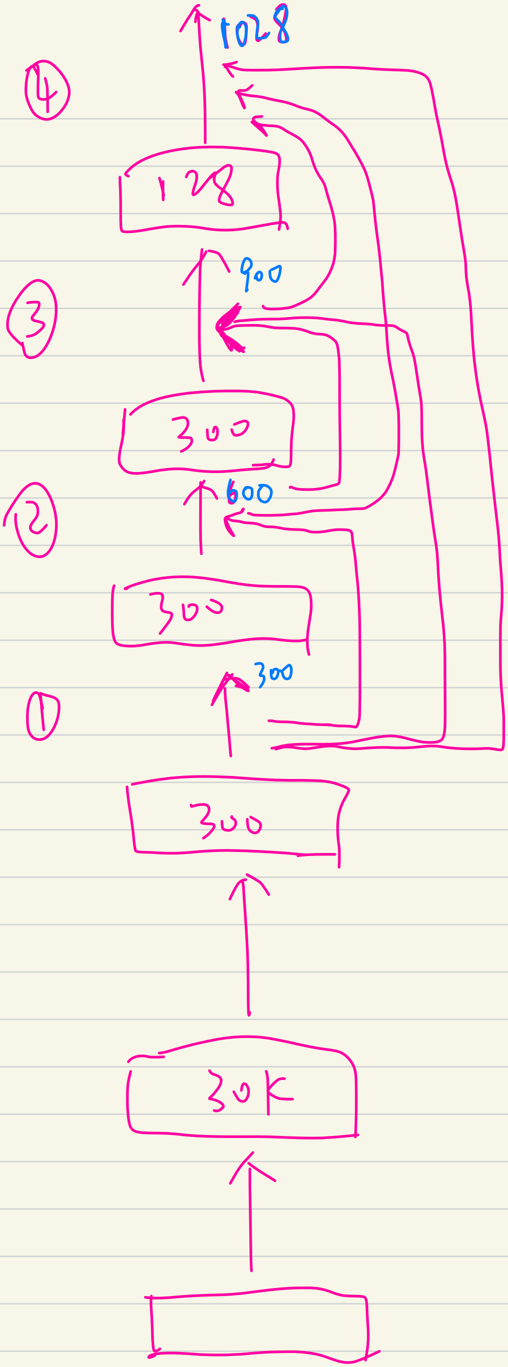


30k → 300

300 → 300

300 → 300

300 → 128

30K → 300

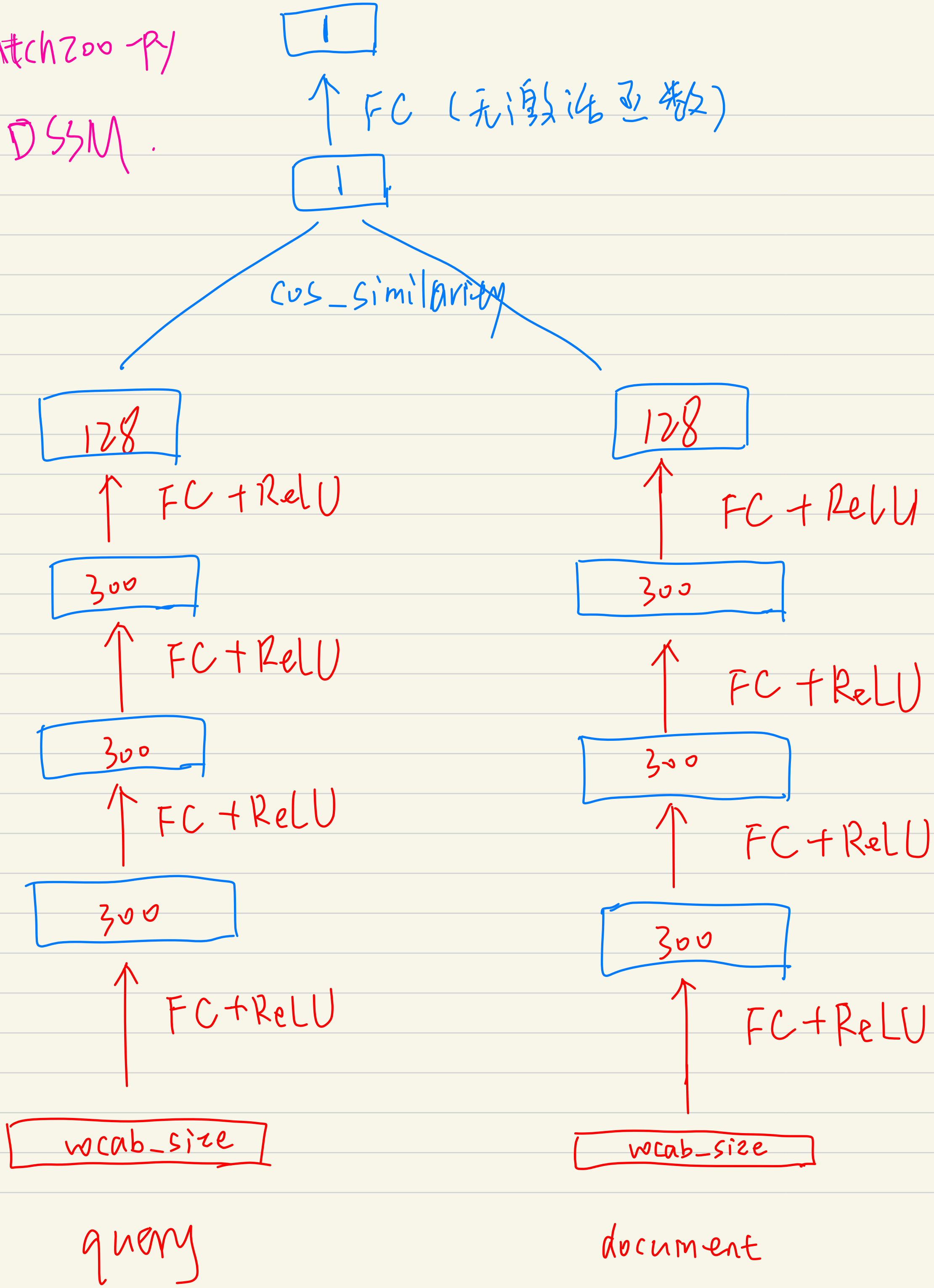
300 → 300

600 → 300

900 → 128

1028 → cos

Matchzoo-Py 的 DSSM.



Learning Deep Structured Semantic Models for Web Search using Clickthrough Data

数据集

Po-Sen Huang
University of Illinois at Urbana-Champaign
405 N Mathews Ave. Urbana, IL 61801 USA
huang146@illinois.edu

Xiaodong He, Jianfeng Gao, Li Deng,
Alex Acero, Larry Heck
Microsoft Research, Redmond, WA 98052 USA
{xiaohe, jfgao, deng, alexac, lheck}@microsoft.com

ABSTRACT

Latent semantic models, such as LSA, intend to map a query to its relevant documents at the semantic level where keyword-based matching often fails. In this study we strive to develop a series of new latent semantic models with a deep structure that project queries and documents into a common low-dimensional space where the relevance of a document given a query is readily computed as the distance between them. The proposed deep structured semantic models are discriminatively trained by maximizing the conditional likelihood of the clicked documents given a query using the clickthrough data. To make our models applicable to large-scale Web search applications, we also use a technique called word hashing, which is shown to effectively scale up our semantic models to handle large vocabularies which are common in such tasks. The new models are evaluated on a Web document ranking task using a real-world data set. Results show that our best model significantly outperforms other latent semantic models, which were considered state-of-the-art in the performance prior to the work presented in this paper.

把query和document映射到共同的低维空间，然后计算两者之间的距离.

word hashing 解决词典过大的问题.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.6 [**Artificial Intelligence**]: *Learning*

General Terms

Algorithms, Experimentation

Keywords

Deep Learning, Semantic Model, Clickthrough Data, Web Search

1. INTRODUCTION

Modern search engines retrieve Web documents mainly by matching keywords in documents with those in search queries. However, lexical matching can be inaccurate due to the fact that a concept is often expressed using different vocabularies and language styles in documents and queries.

Latent semantic models such as latent semantic analysis (LSA) are able to map a query to its relevant documents at the semantic level where lexical matching often fails (e.g., [6][15][2][8][21]). These latent semantic models address the language discrepancy between Web documents and search queries by grouping different terms that occur in a similar context into the same semantic cluster. Thus, a query and a document, represented as two vectors in the lower-dimensional semantic space, can still have a high similarity score even if they do not share any term. Extending from LSA, probabilistic topic models such as probabilistic LSA (PLSA) and Latent Dirichlet Allocation (LDA) have also been proposed for semantic matching [15][2]. However, these models are often trained in an unsupervised manner using an objective function that is only loosely coupled with the evaluation metric for the retrieval task. Thus the performance of these models on Web search tasks is not as good as originally expected.

Recently, two lines of research have been conducted to extend the aforementioned latent semantic models, which will be briefly reviewed below.

First, clickthrough data, which consists of a list of queries and their clicked documents, is exploited for semantic modeling so as to bridge the language discrepancy between search queries and Web documents [9][10]. For example, Gao et al. [10] propose the use of Bi-Lingual Topic Models (BLTMs) and linear Discriminative Projection Models (DPMs) for query-document matching at the semantic level. These models are trained on clickthrough data using objectives that tailor to the document ranking task. More specifically, BLTM is a generative model that requires that a query and its clicked documents not only share the same distribution over topics but also contain similar factions of words assigned to each topic. In contrast, the DPM is learned using the S2Net algorithm [26] that follows the pairwise learning-to-rank paradigm outlined in [3]. After projecting term vectors of queries and documents into concept vectors in a low-dimensional semantic space, the concept vectors of the query and its clicked documents have a smaller distance than that of the query and its unclicked documents. Gao et al. [10] report that both BLTM and DPM outperform significantly the unsupervised latent semantic models, including LSA and PLSA, in the document ranking task. However, the training of BLTM, though using clickthrough data, is to maximize a log-likelihood criterion which is sub-optimal for the evaluation metric for document ranking. On the other hand, the training of DPM involves large-scale matrix multiplications. The sizes of these matrices often grow quickly with the vocabulary size, which could be of an order of millions in Web search tasks. In order to make the training time tolerable, the vocabulary was pruned aggressively. Although a small vocabulary makes the models trainable, it leads to suboptimal performance.

In the second line of research, Salakhutdinov and Hinton extended the semantic modeling using deep auto-encoders [22].

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2263-8/13/10…$15.00.

They demonstrated that hierarchical semantic structure embedded in the query and the document can be extracted via deep learning. Superior performance to the conventional LSA is reported [22]. However, the deep learning approach they used still adopts an unsupervised learning method where the model parameters are optimized for the reconstruction of the documents rather than for differentiating the relevant documents from the irrelevant ones for a given query. As a result, the deep learning models do not significantly outperform the baseline retrieval models based on keyword matching. Moreover, the semantic hashing model also faces the scalability challenge regarding large-scale matrix multiplication. We will show in this paper that the capability of learning semantic models with large vocabularies is crucial to obtain good results in real-world Web search tasks.

In this study, extending from both research lines discussed above, we propose a series of Deep Structured Semantic Models (DSSM) for Web search. More specifically, our best model uses a deep neural network (DNN) to rank a set of documents for a given query as follows. First, a non-linear projection is performed to map the query and the documents to a common semantic space. Then, the relevance of each document given the query is calculated as the cosine similarity between their vectors in that semantic space. The neural network models are discriminatively trained using the clickthrough data such that the conditional likelihood of the clicked document given the query is maximized. Different from the previous latent semantic models that are learned in an unsupervised fashion, our models are optimized directly for Web document ranking, and thus give superior performance, as we will show shortly. Furthermore, to deal with large vocabularies, we propose the so-called *word hashing* method, through which the high-dimensional term vectors of queries or documents are projected to low-dimensional letter based *n*-gram vectors with little information loss. In our experiments, we show that, by adding this extra layer of representation in semantic models, word hashing enables us to learn discriminatively the semantic models with large vocabularies, which are essential for Web search. We evaluated the proposed DSSMs on a Web document ranking task using a real-world data set. The results show that our best model outperforms all the competing methods with a significant margin of 2.5-4.3% in NDCG@1.

In the rest of the paper, Section 2 reviews related work. Section 3 describes our DSSM for Web search. Section 4 presents the experiments, and Section 5 concludes the paper.

2. RELATED WORK

Our work is based on two recent extensions to the latent semantic models for IR. The first is the exploration of the clickthrough data for learning latent semantic models in a supervised fashion [10]. The second is the introduction of deep learning methods for semantic modeling [22].

2.1 Latent Semantic Models and the Use of Clickthrough Data

The use of latent semantic models for query-document matching is a long-standing research topic in the IR community. Popular models can be grouped into two categories, linear projection models and generative topic models, which we will review in turn.

The most well-known linear projection model for IR is LSA [6]. By using the singular value decomposition (SVD) of a document-term matrix, a document (or a query) can be mapped to a low-dimensional concept vector $\widehat{\mathbf{D}} = \mathbf{A}^T\mathbf{D}$, where the $\mathbf{A}$ is the projection matrix. In document search, the relevance score between a query and a document, represented respectively by term vectors $\mathbf{Q}$ and $\mathbf{D}$, is assumed to be proportional to their cosine similarity score of the corresponding concept vectors $\widehat{\mathbf{Q}}$ and $\widehat{\mathbf{D}}$, according to the projection matrix $\mathbf{A}$

$$\mathrm{sim}_{\mathbf{A}}(\mathbf{Q}, \mathbf{D}) = \frac{\widehat{\mathbf{Q}}^T\widehat{\mathbf{D}}}{\|\widehat{\mathbf{Q}}\|\|\widehat{\mathbf{D}}\|} \qquad (1)$$

In addition to latent semantic models, the translation models trained on clicked query-document pairs provide an alternative approach to semantic matching [9]. Unlike latent semantic models, the translation-based approach learns translation relationships directly between a term in a document and a term in a query. Recent studies show that given large amounts of clickthrough data for training, this approach can be very effective [9][10]. We will also compare our approach with translation models experimentally as reported in Section 4.

2.2 Deep Learning

Recently, deep learning methods have been successfully applied to a variety of language and information retrieval applications [1][4][7][19][22][23][25]. By exploiting deep architectures, deep learning techniques are able to discover from training data the hidden structures and features at different levels of abstractions useful for the tasks. In [22] Salakhutdinov and Hinton extended the LSA model by using a deep network (auto-encoder) to discover the hierarchical semantic structure embedded in the query and the document. They proposed a semantic hashing (SH) method which uses bottleneck features learned from the deep auto-encoder for information retrieval. These deep models are learned in two stages. First, a stack of generative models (i.e., the restricted Boltzmann machine) are learned to map layer-by-layer a term vector representation of a document to a low-dimensional semantic concept vector. Second, the model parameters are fine-tuned so as to minimize the cross entropy error between the original term vector of the document and the reconstructed term vector. The intermediate layer activations are used as features (i.e., bottleneck) for document ranking. Their evaluation shows that the SH approach achieves a superior document retrieval performance to the LSA. However, SH suffers from two problems, and cannot outperform the standard lexical matching based retrieval model (e.g., cosine similarity using TF-IDF term weighting). The first problem is that the model parameters are optimized for the re-construction of the document term vectors rather than for differentiating the relevant documents from the irrelevant ones for a given query. Second, in order to make the computational cost manageable, the term vectors of documents consist of only the most-frequent 2000 words. In the next section, we will show our solutions to these two problems.

3. DEEP STRUCTURED SEMANTIC MODELS FOR WEB SEARCH

3.1 DNN for Computing Semantic Features

The typical DNN architecture we have developed for mapping the raw text features into the features in a semantic space is shown in Fig. 1. The input (raw text features) to the DNN is a high-dimensional term vector, e.g., raw counts of terms in a query or a document without normalization, and the output of the DNN is a concept vector in a low-dimensional semantic feature space. This

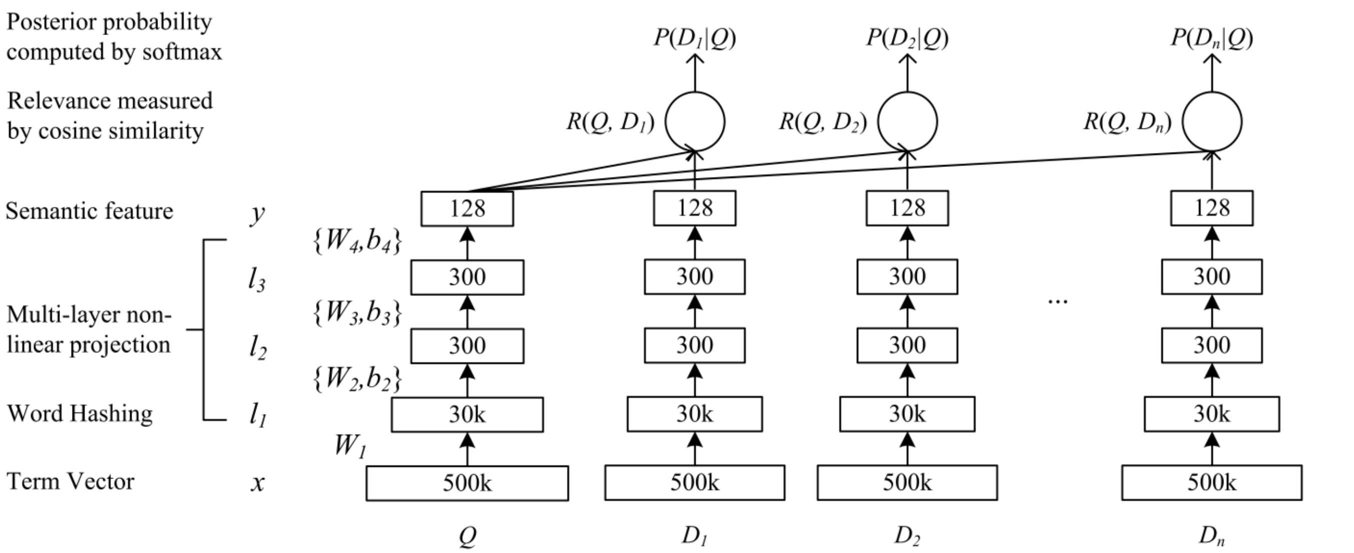


Figure 1: Illustration of the DSSM. It uses a DNN to map high-dimensional sparse text features into low-dimensional dense features in a semantic space. The first hidden layer, with 30k units, accomplishes word hashing. The word-hashed features are then projected through multiple layers of non-linear projections. The final layer's neural activities in this DNN form the feature in the semantic space.

DNN model is used for Web document ranking as follows: 1) to map term vectors to their corresponding semantic concept vectors; 2) to compute the relevance score between a document and a query as cosine similarity of their corresponding semantic concept vectors; rf. Eq. (3) to (5).

整体思路：
①、映射到隐性向量
②.计算 cos similarity

More formally, if we denote x as the input term vector, y as the output vector, l_i, $i = 1, \dots, N-1$, as the intermediate hidden layers, W_i as the i-th weight matrix, and b_i as the i-th bias term, we have

$$\begin{aligned} l_1 &= W_1 x \\ l_i &= f(W_i l_{i-1} + b_i), i = 2, \dots, N-1 \\ y &= f(W_N l_{N-1} + b_N) \end{aligned} \tag{3}$$

激活函数
用 tanh

where we use the $tanh$ as the activation function at the output layer and the hidden layers $l_i, i = 2, \dots, N-1$:

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{4}$$

The semantic relevance score between a query Q and a document D is then measured as:

$$R(Q, D) = \text{cosine}(y_Q, y_D) = \frac{{y_Q}^T y_D}{\|y_Q\|\|y_D\|} \tag{5}$$

cos similarity

where y_Q and y_D are the concept vectors of the query and the document, respectively. In Web search, given the query, the documents are sorted by their semantic relevance scores.

Conventionally, the size of the term vector, which can be viewed as the raw bag-of-words features in IR, is identical to that of the vocabulary that is used for indexing the Web document collection. The vocabulary size is usually very large in real-world Web search tasks. Therefore, when using term vector as the input, the size of the input layer of the neural network would be unmanageable for inference and model training. To address this problem, we have developed a method called "word hashing" for the first layer of the DNN, as indicated in the lower portion of Figure 1. This layer consists of only linear hidden units in which the weight matrix of a very large size is not learned. In the following section, we describe the word hashing method in detail.

3.2 Word Hashing

The word hashing method described here aim to reduce the dimensionality of the bag-of-words term vectors. It is based on letter n-gram, and is a new method developed especially for our task. Given a word (e.g. *good*), we first add word starting and ending marks to the word (e.g. *#good#*). Then, we break the word into letter n-grams (e.g. letter trigrams: *#go*, *goo*, *ood*, *od#*). Finally, the word is represented using a vector of letter n-grams.

One problem of this method is *collision*, i.e., two different words could have the same letter n-gram vector representation. Table 1 shows some statistics of word hashing on two vocabularies. Compared with the original size of the one-hot vector, word hashing allows us to represent a query or a document using a vector with much lower dimensionality. Take the 40K-word vocabulary as an example. Each word can be represented by a 10,306-dimentional vector using letter trigrams, giving a four-fold dimensionality reduction with few collisions. The reduction of dimensionality is even more significant when the technique is applied to a larger vocabulary. As shown in Table 1, each word in the 500K-word vocabulary can be represented by a 30,621 dimensional vector using letter trigrams, a reduction of 16-fold in dimensionality with a negligible collision rate of 0.0044% (22/500,000).

两个不同的词可能有相同的 n-gram
向量表达

While the number of English words can be unlimited, the number of letter n-grams in English (or other similar languages) is often limited. Moreover, word hashing is able to map the morphological variations of the same word to the points that are close to each other in the letter n-gram space. More importantly, while a word unseen in the training set always cause difficulties in word-based representations, it is not the case where the letter n-gram based representation is used. The only risk is the minor representation collision as quantified in Table 1. Thus, letter n-gram based word hashing is robust to the out-of-vocabulary problem, allowing us to scale up the DNN solution to the Web search tasks where extremely large vocabularies are desirable. We will demonstrate the benefit of the technique in Section 4.

word hash
可以把词的
不同变体映
射到比较接
近 n-gram 空间

word hashing
对 OOV 问题
更鲁棒，也更
适用于 web
data 词典
过大的问题

In our implementation, the letter n-gram based word hashing can be viewed as a fixed (i.e., non-adaptive) linear transformation,

word hash 可以视为一种线性变换

through which an term vector in the input layer is projected to a letter *n*-gram vector in the next layer higher up, as shown in Figure 1. Since the letter *n*-gram vector is of a much lower dimensionality, DNN learning can be carried out effectively.

	Letter-Bigram		Letter-Trigram	
Word Size	**Token Size**	**Collision**	**Token Size**	**Collision**
40k	1107	18	10306	2
500k	1607	1192	30621	22

Table 1: Word hashing token size and collision numbers as a function of the vocabulary size and the type of letter ngrams.

3.3 Learning the DSSM

The clickthrough logs consist of a list of queries and their clicked documents. We assume that a query is relevant, at least partially, to the documents that are clicked on for that query. Inspired by the discriminative training approaches in speech and language processing , we thus propose a supervised training method to learn our model parameters, i.e., the weight matrices W_i and bias vectors b_i in our neural network as the essential part of the DSSM, so as to maximize the conditional likelihood of the clicked documents given the queries.

First, we compute the posterior probability of a document given a query from the semantic relevance score between them through a softmax function

包括正例 + 4个随机采样的负例·

$$P(D|Q) = \frac{\exp(\gamma R(Q,D))}{\sum_{D' \in \boldsymbol{D}} \exp(\gamma R(Q,D'))} \tag{6}$$

where γ is a smoothing factor in the softmax function, which is set empirically on a held-out data set in our experiment. $\boldsymbol{D}$ denotes the set of candidate documents to be ranked. Ideally, $\boldsymbol{D}$ should contain all possible documents. In practice, for each (query, clicked-document) pair, denoted by (Q, D^+) where Q is a query and D^+ is the clicked document, we approximate $\boldsymbol{D}$ by including D^+ and four randomly selected unclicked documents, denote by $\{D_j^-; j = 1, \ldots, 4\}$. In our pilot study, we do not observe any significant difference when different sampling strategies were used to select the unclicked documents.

In training, the model parameters are estimated to maximize the likelihood of the clicked documents given the queries across the training set. Equivalently, we need to minimize the following loss function

$$L(\Lambda) = -\log \prod_{(Q,D^+)} P(D^+|Q) \tag{7}$$

where Λ denotes the parameter set of the neural networks $\{W_i, b_i\}$. Since $L(\Lambda)$ is differentiable w.r.t. to Λ, the model is trained readily using gradient-based numerical optimization algorithms. The detailed derivation is omitted due to the space limitation.

3.4 Implementation Details

To determine the training parameters and to avoid over-fitting, we divided the clickthrough data into two factions that do not overlap, called training and validation datasets, respectively. In our experiments, the models are trained on the training set and the training parameters are optimized on the validation dataset. For the DNN experiments, we used the architecture with three hidden layers as shown in Figure 1. The first hidden layer is the word hashing layer containing about 30k nodes (e.g., the size of the letter-trigrams as shown in Table 1). The next two hidden layers have 300 hidden nodes each, and the output layer has 128 nodes. Word hashing is based on a fixed projection matrix. The similarity measure is based on the output layer with the dimensionality of 128. Following [20], we initialize the network weights with uniform distribution in the range between $-\sqrt{6/(fanin + fanout)}$ and $\sqrt{6/(fanin + fanout)}$ where $fanin$ and $fanout$ are the number of input and output units, respectively. Empirically, we have not observed better performance by doing layer-wise pre-training. In the training stage, we optimize the model using mini-batch based stochastic gradient descent (SGD). Each mini-batch consists of 1024 training samples. We observed that the DNN training usually converges within 20 epochs (passes) over the entire training data.

参数初始化

batch size

4. EXPERIMENTS

We evaluated the DSSM, proposed in Section 3, on the Web document ranking task using a real-world data set. In this section, we first describe the data set on which the models are evaluated. Then, we compare the performances of our best model against other state of the art ranking models. We also investigate the break-down impact of the techniques proposed in Section 3.

4.1 Data Sets and Evaluation Methodology

We have evaluated the retrieval models on a large-scale real world data set, called the evaluation data set henceforth. The evaluation data set contains 16,510 English queries sampled from one-year query log files of a commercial search engine. On average, each query is associated with 15 Web documents (URLs). Each query-title pair has a relevance label. The label is human generated and is on a 5-level relevance scale, 0 to 4, where level 4 means that the document is the most relevant to query Q and 0 means D is not relevant to Q. All the queries and documents are preprocessed such that the text is white-space tokenized and lowercased, numbers are retained, and no stemming/inflection is performed.

All ranking models used in this study (i.e., DSSM, topic models, and linear projection models) contain many free hyper-parameters that must be estimated empirically. In all experiments, we have used 2-fold cross validation: A set of results on one half of the data is obtained using the parameter settings optimized on the other half, and the global retrieval results are combined from the two sets.

The performance of all ranking models we have evaluated has been measured by mean Normalized Discounted Cumulative Gain (NDCG) [17], and we will report NDCG scores at truncation levels 1, 3, and 10 in this section. We have also performed a significance test using the paired t-test. Differences are considered statistically significant when the *p*-value is less than 0.05.

In our experiments, we assume that a query is parallel to the titles of the documents clicked on for that query. We extracted large amounts of the query-title pairs for model training from one year query log files using a procedure similar to [11]. Some previous studies, e.g., [24][11], showed that the query click field, when it is valid, is the most effective piece of information for Web search, seconded by the title field. However, click information is unavailable for many URLs, especially new URLs and tail URLs, leaving their click fields invalid (i.e., the field is either empty or unreliable because of sparseness). In this study, we assume that each document contained in the evaluation data set is either a new URL or a tail URL, thus has no click information (i.e., its click field is invalid). Our research goal is to investigate how to learn the latent semantic models from the popular URLs that have rich click information, and apply the models to improve the retrieval of those tail or new URLs. To this end, in our experiments only the title fields of the Web documents are used for ranking. For training latent semantic models, we use a randomly sampled subset of approximately 100 million pairs whose documents are popular and have rich click information. We then test trained models in ranking the documents in the evaluation data set containing no click information. The query-title pairs are pre-processed in the same way as the evaluation data to ensure uniformity.

4.2 Results

The main results of our experiments are summarized in Table 2, where we compared our best version of the DSSM (Row 12) with three sets of baseline models. The first set of baselines includes a couple of widely used lexical matching methods such as TF-IDF (Row 1) and BM25 (Row 2). The second is a word translation model (WTM in Row 3) which is intended to directly address the query-document language discrepancy problem by learning a lexical mapping between query words and document words [9][10]. The third includes a set of state-of-the-art latent semantic models which are learned either on documents only in an unsupervised manner (LSA, PLSA, DAE as in Rows 4 to 6) or on clickthrough data in a supervised way (BLTM-PR, DPM, as in Rows 7 and 8). In order to make the results comparable, we re-implement these models following the descriptions in [10], e.g., models of LSA and DPM are trained using a 40k-word vocabulary due to the model complexity constraint, and the other models are trained using a 500K-word vocabulary. Details are elaborated in the following paragraphs.

TF-IDF (Row 1) is the baseline model, where both documents and queries represented as term vectors with TF-IDF term weighting. The documents are ranked by the cosine similarity between the query and document vectors. We also use **BM25** (Row 2) ranking model as one of our baselines. Both **TF-IDF** and **BM25** are state-of-the-art document ranking models based on term matching. They have been widely used as baselines in related studies.

WTM (Rows 3) is our implementation of the word translation model described in [9], listed here for comparison. We see that WTM outperforms both baselines (TF-IDF and BM25) significantly, confirming the conclusion reached in [9]. **LSA** (Row 4) is our implementation of latent semantic analysis model. We used PCA instead of SVD to compute the linear projection matrix. Queries and titles are treated as separate documents, the pair information from the clickthrough data was not used in this model. **PLSA** (Rows 5) is our implementation of the model proposed in [15], and was trained on documents only (i.e., the title side of the query-title pairs). Different from [15], our version of PLSA was learned using MAP estimation as in [10]. **DAE** (Row 6) is our implementation of the deep auto-encoder based semantic hashing model proposed by Salakhutdinov and Hinton in [22]. Due to the model training complexity, the input term vector is based on a 40k-word vocabulary. The DAE architecture contains four hidden layers, each of which has 300 nodes, and a bottleneck layer in the middle which has 128 nodes. The model is trained on documents only in an unsupervised manner. In the fine-tuning stage, we used cross-entropy error as training criteria. The central layer activations are used as features for the computation of cosine similarity between query and document. Our results are consistent with previous results reported in [22]. The DNN based latent semantic model outperforms the linear projection model (e.g., LSA). However, both LSA and DAE are trained in an unsupervised fashion on document collection only, thus cannot outperform the state-of-the-art lexical matching ranking models.

BLTM-PR (Rows 7) is the best performer among different versions of the bilingual topic models described in [10]. BLTM with posterior regularization (BLTM-PR) is trained on query-title pairs using the EM algorithm with a constraint enforcing the paired query and title to have same fractions of terms assigned to each hidden topic. **DPM** (Row 8) is the linear discriminative projection model proposed in [10], where the projection matrix is discriminatively learned using the S2Net algorithm [26] on relevant and irrelevant pairs of queries and titles. Similar to that BLTM is an extension to PLSA, DPM can also be viewed as an extension of LSA, where the linear projection matrix is learned in a supervised manner using clickthrough data, optimized for document ranking. We see that using clickthrough data for model training leads to some significant improvement. Both BLTM-PR and DPM outperform the baseline models (TF-IDF and BM25).

Rows 9 to 12 present results of different settings of the DSSM. **DNN** (Row 9) is a DSSM without using word hashing. It uses the same structure as DAE (Row 6), but is trained in a supervised fashion on the clickthrough data. The input term vector is based on a 40k-word vocabulary, as used by DAE. **L-WH linear** (Row 10) is the model built using letter trigram based word hashing and supervised training. It differs from the **L-WH non-linear** model (Row 11) in that we do not apply any nonlinear activation function, such as *tanh*, to its output layer. **L-WH DNN** (Row 12) is our best DNN-based semantic model, which uses three hidden layers, including the layer with the Letter-trigram-based Word Hashing (L-WH), and an output layer, and is discriminatively trained on query-title pairs, as described in Section 3. Although the letter *n*-gram based word hashing method can be applied to arbitrarily large vocabularies, in order to perform a fair comparison with other competing methods, the model uses a 500K-word vocabulary.

The results in Table 2 show that the deep structured semantic model is the best performer, beating other methods by a statistically significant margin in NDCG and demonstrating the empirical effectiveness of using DNNs for semantic matching.

From the results in Table 2, it is also clear that supervised learning on clickthrough data, coupled with an IR-centric optimization criterion tailoring to ranking, is essential for obtaining superior document ranking performance. For example, both DNN and DAE (Row 9 and 6) use the same 40k-word vocabulary and adopt the same deep architecture. The former outperforms the latter by 3.2 points in NDCG@1.

Word hashing allows us to use very large vocabularies for modeling. For instance, the models in Rows 12, which use a 500k-word vocabulary (with word hashing), significantly outperform the model in Row 9, which uses a 40k-word vocabulary, although the former has slightly fewer free parameters than the later since the word hashing layer containing about only 30k nodes.

We also evaluated the impact of using a deep architecture versus a shallow one in modeling semantic information embedded in a query and a document. Results in Table 2 show that DAE (Row 3) is better than LSA (Row 2), while both LSA and DAE are unsupervised models. We also have observed similar results when comparing the shallow vs. deep architecture in the case of supervised models. Comparing models in Rows 11 and 12 respectively, we observe that increasing the number of nonlinear layers from one to three raises the NDCG scores by 0.4-0.5 point which are statistically significant, while there is no significant difference between linear and non-linear models if both are one-layer shallow models (Row 10 vs. Row 11).

#	Models	NDCG@1	NDCG@3	NDCG@10
1	TF-IDF	0.319	0.382	0.462
2	BM25	0.308	0.373	0.455
3	WTM	0.332	0.400	0.478
4	LSA	0.298	0.372	0.455
5	PLSA	0.295	0.371	0.456
6	DAE	0.310	0.377	0.459
7	BLTM-PR	0.337	0.403	0.480
8	DPM	0.329	0.401	0.479
9	DNN	0.342	0.410	0.486
10	L-WH linear	0.357	0.422	0.495
11	L-WH non-linear	0.357	0.421	0.494
12	**L-WH DNN**	**0.362**	**0.425**	**0.498**

Table 2: Comparative results with the previous state of the art approaches and various settings of DSSM.

5. CONCLUSIONS

We present and evaluate a series of new latent semantic models, notably those with deep architectures which we call the DSSM. The main contribution lies in our significant extension of the previous latent semantic models (e.g., LSA) in three key aspects. First, we make use of the clickthrough data to optimize the parameters of all versions of the models by directly targeting the goal of document ranking. Second, inspired by the deep learning framework recently shown to be highly successful in speech recognition [5][13][14][16][18], we extend the linear semantic models to their nonlinear counterparts using multiple hidden-representation layers as. The deep architectures adopted have further enhanced the modeling capacity so that more sophisticated semantic structures in queries and documents can be captured and represented. Third, we use a letter n-gram based word hashing technique that proves instrumental in scaling up the training of the deep models so that very large vocabularies can be used in realistic web search. In our experiments, we show that the new techniques pertaining to each of the above three aspects lead to significant performance improvement on the document ranking task. A combination of all three sets of new techniques has led to a new state-of-the-art semantic model that beats all the previously developed competing models with a significant margin.

REFERENCES

[1] Bengio, Y., 2009. “Learning deep architectures for AI.” Foundumental Trends Machine Learning, vol. 2.

[2] Blei, D. M., Ng, A. Y., and Jordan, M. J. 2003. “Latent Dirichlet allocation.” In JMLR, vol. 3.

[3] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, and Hullender, G. 2005. “Learning to rank using gradient descent.” In ICML.

[4] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P., 2011. “Natural language processing (almost) from scratch.” in JMLR, vol. 12.

[5] Dahl, G., Yu, D., Deng, L., and Acero, A., 2012. “Context-dependent pre-trained deep neural networks for large vocabulary speech recognition.” in IEEE Transactions on Audio, Speech, and Language Processing.

[6] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T., and Harshman, R. 1990. “Indexing by latent semantic analysis.” J. American Society for Information Science, 41(6): 391-407

[7] Deng, L., He, X., and Gao, J., 2013. "Deep stacking networks for information retrieval." In ICASSP

[8] 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.

[9] Gao, J., He, X., and Nie, J-Y. 2010. “Clickthrough-based translation models for web search: from word models to phrase models.” In CIKM.

[10] Gao, J., Toutanova, K., Yih., W-T. 2011. “Clickthrough-based latent semantic models for web search.” In SIGIR.

[11] Gao, J., Yuan, W., Li, X., Deng, K., and Nie, J-Y. 2009. “Smoothing clickthrough data for web search ranking.” In SIGIR.

[12] He, X., Deng, L., and Chou, W., 2008. “Discriminative learning in sequential pattern recognition,” Sept. IEEE Sig. Proc. Mag.

[13] Heck, L., Konig, Y., Sonmez, M. K., and Weintraub, M. 2000. “Robustness to telephone handset distortion in speaker recognition by discriminative feature design.” In Speech Communication.

[14] 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 Sig. Proc. Mag.

[15] Hofmann, T. 1999. “Probabilistic latent semantic indexing.” In SIGIR.

[16] Hutchinson, B., Deng, L., and Yu, D., 2013. “Tensor deep stacking networks.” In IEEE T-PAMI, vol. 35.

[17] Jarvelin, K. and Kekalainen, J. 2000. “IR evaluation methods for retrieving highly relevant documents.” In SIGIR.

[18] Konig, Y., Heck, L., Weintraub, M., and Sonmez, M. K. 1998. “Nonlinear discriminant feature extraction for robust text-independent speaker recognition.” in RLA2C.

[19] Mesnil, G., He, X., Deng, L., and Bengio, Y., 2013. “Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding.” In Interspeech.

[20] Montavon, G., Orr, G., Müller, K., 2012. Neural Networks: Tricks of the Trade (Second edition). Springer.

[21] Platt, J., Toutanova, K., and Yih, W. 2010. “Translingual doc-ument representations from discriminative projections.” In EMNLP.

[22] Salakhutdinov R., and Hinton, G., 2007 “Semantic hashing.” in Proc. SIGIR Workshop Information Retrieval and Applications of Graphical Models.

[23] Socher, R., Huval, B., Manning, C., Ng, A., 2012. “Semantic compositionality through recursive matrix-vector spaces.” In EMNLP.

[24] Svore, K., and Burges, C. 2009. “A machine learning approach for improved BM25 retrieval.” In CIKM.

[25] Tur, G., Deng, L., Hakkani-Tur, D., and He, X., 2012. “Towards deeper understanding deep convex networks for semantic utterance classification.” In ICASSP.

[26] Yih, W., Toutanova, K., Platt, J., and Meek, C. 2011. “Learning discriminative projections for text similarity measures.” In CoNLL.

APPENDIX

I. Gradient Computation and Gradient Descent

Since $L(\Lambda)$ is differentiable w.r.t. to Λ, the model is trained readily using gradient-based numerical optimization algorithms. The update rule is

$$\Lambda_t = \Lambda_{t-1} - \epsilon_t \frac{\partial L(\Lambda)}{\partial \Lambda}|_{\Lambda=\Lambda_{t-1}} \tag{8}$$

where ϵ_t is the learning rate at the t^{th} iteration, Λ_t and Λ_{t-1} are the models at the t^{th} and the $(t-1)^{th}$ iteration, respectively.

In what follows, we derive the gradient of the loss function w.r.t. the parameters of the neural networks. Assuming that there are in total R (query, clicked-document) pairs, we denote (Q_r, D_r^+) as the r-th (query, clicked-document) pair. Then, if we denote

$$L_r(\Lambda) = -\log P(D_r^+|Q_r) \tag{9}$$

we have

$$\frac{\partial L(\Lambda)}{\partial \Lambda} = \sum_{r=1}^{R} \frac{\partial L_r(\Lambda)}{\partial \Lambda} \tag{10}$$

In the following, we will show the derivation of $\frac{\partial L_r(\Lambda)}{\partial \Lambda}$.

For a query Q and a document D, we denote $l_{i,Q}$ and $l_{i,D}$ be the activation in the hidden layer i, and y_Q and y_D be the output activation for Q and Q , respectively. They are computed according to Eq. (3).

We then derive $\frac{\partial L_r(\Lambda)}{\partial \Lambda}$ as follows[1]. For simplification, the subscript of r will be omitted hereafter.

First, the loss function in Eq. (9) can be written as:

$$L(\Lambda) = \log\left(1 + \sum\nolimits_j \exp(-\gamma\,\Delta_j)\right) \tag{11}$$

where $\Delta_j = R(Q, D^+) - R(Q, D_j^-)$. The gradient of the loss function w.r.t. the N-th weight matrix W_N is

$$\frac{\partial L(\Lambda)}{\partial W_N} = \sum_j \alpha_j \frac{\partial \Delta_j}{\partial W_N} \tag{12}$$

where

$$\frac{\partial \Delta_j}{\partial W_N} = \frac{\partial R(Q, D^+)}{\partial W_N} - \frac{\partial R(Q, D_j^-)}{\partial W_N} \tag{13}$$

and

$$\alpha_j = \frac{-\gamma \exp(-\gamma\,\Delta_j)}{1 + \sum_{j'} \exp(-\gamma\,\Delta_{j'})} \tag{14}$$

To simplify the notation, let a, b, c be ${y_Q}^T y_D$, $1/\|y_Q\|$, and $1/\|y_D\|$, respectively. With $tanh$ as the activation function in our model, each term in the right-hand side of Eq. (13) can be calculated using the following formula:

$$\frac{\partial R(Q, D)}{\partial W_N} = \frac{\partial}{\partial W_N} \frac{{y_Q}^T y_D}{\|y_Q\|\|y_D\|} = \delta_{y_Q}^{(Q,D)} l_{N-1,Q}^T + \delta_{y_D}^{(Q,D)} l_{N-1,D}^T \tag{15}$$

where $\delta_{y_Q}^{(Q,D)}$ and $\delta_{y_D}^{(Q,D)}$ for a pair of (Q, D) are computed as

$$\begin{aligned} \delta_{y_Q}^{(Q,D)} &= (1 - y_Q) \circ (1 + y_Q) \circ (bcy_D - acb^3 y_Q) \\ \delta_{y_D}^{(Q,D)} &= (1 - y_D) \circ (1 + y_D) \circ (bcy_Q - abc^3 y_D) \end{aligned} \tag{16}$$

where the operator $\circ$ is the element-wise multiplication (Hadamard product).

For hidden layers, we also need to calculate $\{\delta\}$ for each Δ_j. For example, each δ in the hidden layer i can be calculated through back propagation as

$$\begin{aligned} \delta_{i,Q}^{(Q,D)} &= (1 + l_{i,Q}) \circ (1 - l_{i,Q}) \circ W_{i+1}^T \delta_{i+1,Q}^{(Q,D)} \\ \delta_{i,D}^{(Q,D)} &= (1 + l_{i,D}) \circ (1 - l_{i,D}) \circ W_{i+1}^T \delta_{i+1,D}^{(Q,D)} \end{aligned} \tag{17}$$

and eventually we have $\delta_{N,Q}^{(Q,D)} = \delta_{y_Q}^{(Q,D)}$ and $\delta_{N,D}^{(Q,D)} = \delta_{y_D}^{(Q,D)}$.

Correspondingly, the gradient of the loss function w.r.t. the intermediate weight matrix, $W_i, i = 2, \ldots, N-1$, can be computed as[2]

$$\frac{\partial L(\Lambda)}{\partial W_i} = \sum_j \alpha_j \frac{\partial \Delta_j}{\partial W_i} \tag{18}$$

where

$$\begin{aligned} \frac{\partial \Delta_j}{\partial W_i} = &\left(\delta_{i,Q}^{(Q,D^+)} l_{i-1,Q}^T + \delta_{i,D^+}^{(Q,D^+)} l_{i-1,D^+}^T\right) \\ &- \left(\delta_{i,Q}^{(Q,D_j^-)} l_{i-1,Q}^T + \delta_{i,D_j^-}^{(Q,D_j^-)} l_{i-1,D_j^-}^T\right) \end{aligned} \tag{19}$$

II. Analysis on Document Ranking Errors

In the test data, among 16,412 unique queries, we compare each query's NDCG@1 values using TF-IDF and our best model, letter trigram based word hashing with supervised DNN (L-WH DNN). There are in total 1,985 queries on which L-WH DNN performs better than TF-IDF (the sum of NDCG@1 differences is1332.3). On the other hand, TF-IDF outperforms L-WH DNN on 1077 queries (the sum of NDCG@1 differences is 630.61). For both cases, we sample several concrete examples. They are shown in Tables 5 and 6, respectively. We observe in Table 5 that the NDCG improvement is largely to the better match between queries and titles in the semantic level than in the lexical level.

[1] We present only the derivation for the weight matrices. The derivation for the bias vector is similar and is omitted.

[2] Note that W_1 is the matrix of word hashing. It is fixed and need no training.

	L-WH DNN wins over TF-IDF	
	Query	Title
1	bfpo	postcodes in the united kingdom wikipedia the free encyclopedia
2	univ of penn	university of pennsylvania wikipedia the free encyclopedia
3	citibank	citi com
4	ccra	canada revenue agency website
5	search galleries	photography community including forums reviews and galleries from photo net
6	met art	metropolitan museum of art wikipedia the free encyclopedia
7	new york brides	long island bride and groom wedding magazine website
8	motocycle loans	auto financing is easy with the capital one blank check
9	boat	new and used yarts for sale yachartworld com
10	bbc games	bbc sport

Table 5: Examples that our deep semantic model performs better than TF-IDF.

To make our method more intuitive, we have also visualized the learned hidden representations of the words in the queries and documents. We do so by treating each word as a unique document and passing it as an input to the trained DNN. At each output node, we group all the words with high activation levels and cluster them accordingly. Table 7 shows some example clusters, each corresponding to an output node of the DNN model. It is interesting to see that words with the same or related semantic meanings do stay in the same cluster.

	L-WH DNN loses to TF-IDF	
	Query	Title
1	hey arnold	hey arnold the movie
2	internet by dell	dell hyperconnect mobile internet solutions dell
3	www mcdonalds com	mcdonald s'
4	m t	m t bank
5	board of directors	board of directors west s encyclopedia of american law full ariticle from answers com
6	puppet skits	skits
7	montreal canada attractions	go montreal tourist information
8	how to address a cover letter	how to write a cover letter
9	bbc television	bbc academy
10	rock com	rock music information from answers com

Table 6: Examples that our deep semantic model performs worse than TF-IDF.

automotive	chevrolet	youtube	bear	systems
wheels	fuel	videos	hunting	protect
cars	motorcycle	dvd	texas	platform
auto	toyota	downloads	colorado	efficiency
car	chevy	movie	hunter	oems
vehicle	motorcycles	cd	tucson	systems32

Table 7: Examples of the clustered words on five different output nodes of the trained DNN. The clustering criterion is high activation levels at the output nodes of the DNN.