Building Deep Structured Semantic Similarity Features to Improve the Media Search

Xugang Ye, Zijie Qi, Xiaodong He
Microsoft
{xugangye, xiaohe, zijieqi}@microsoft.com
Jingjing Li
University of Virginia
jl9rf@virginia.edu

Abstract
In media search, it’s often found that for a query, a content without any shared term in its title or concise description can still be highly relevant. Therefore, document retrieval based on only word matching can miss important contents. On the other hand, even with an exact-word match in its document, the content can still be an unsatisfactory result. These observations suggest that query-document matching should be established at the semantic level other than the term level. In this paper, we introduce some latest results of applying the Deep Structured Semantic Model (DSSM) to build the similarity features for document retrieval and ranking in media search. To test the new features, we perform the experiments of retrieving and ranking media documents from Xbox, with and without the DSSM involved. To have a DSSM, we leverage the Bing’s large web search logs to train a seed DSSM and then tune the model using the Xbox’s ranker training data. We also train the Xbox’s rankers with and without adding the DSSM similarity features. Our experimental results have shown that adding the DSSM similarity features significantly improves the results of document retrieval and ranking.

1. Introduction
Applying semantic matching based methods beyond lexical matching based ones to search problems is currently a hot research direction. Latent semantic models such as latent semantic analysis (LSA) have been widely used to find the relatedness of a query and its relevant documents when term matching fails [1][4][6]. Besides LSA, probabilistic topic models such as the probabilistic LSA (PLSA) [6] and Latent Dirichlet Allocation (LDA) [1] have also been proposed for semantic matching. Those models, despite the purpose and solid theoretical foundation, suffer from a fact that they are usually trained via unsupervised learning under the objective functions that are loosely connected to the relevance labels. Consequently, the semantic matching scores that are generated from these models, when used for document retrieval and ranking, have not been seen to have performance as good as originally expected.

On the other hand, one recent semantic matching method called Deep Structured Semantic Model (DSSM) was proposed by Huang et al. [7] in a form of supervised learning, which utilizes the relevance labels in the model training. With the labels incorporated into its objective function, the DSSM maps a query and a relevant document into a common semantic space in which a similarity measure can be calculated so that even there is no shared term between the query and the document, the similarity score can still be positive. In this sense, the DSSM can do better in reducing the chance that relevant documents are missed during the retrieval. Furthermore, it has been reported in [7][9] that applying the DSSM to web search yields superior single feature performance. That is the DSSM similarity score is better than many traditional features including TF-IDF, BM25, WTM (word translation model), LSA, PLSA, and LDA, etc. One important reason is that the DSSM similarity score contains the label information so that not only high relevance without term match receives high similarity score, the low relevance with term match also has low similarity score as a result of penalization.

Given the encouraging results reported so far, we aim to apply the DSSM model to the media search area to see whether the method can also improve the quality of the retrieval and ranking results in the media domain. The rest of this paper is organized as the follows. We first introduce the formulation of the DSSM model. We then introduce letter-tri-gram as the technique of dimension-reduced text representation, the parameter estimation method, the model structure, and how the model is integrated into a media search system. We make conclusions after presenting some experimental results on the Xbox’s media search data.

2. Model

2.1. Formulation
Let \( \{(q_1, d_1), (q_2, d_2), \ldots \} \) denote a set of relevant query-document pairs in which \( d_i \) is a document under the query \( q_i \). Assuming that these relevant query-document pairs are independent, then the joint probability of observing these query-document pairs is
\[
P(d|q) = \prod_i P(d_i|q_i),
\]  
(1)
where \( P(d_i|q_i) \) denotes the conditional probability that \( d_i \) is relevant under \( q_i \). To parameterize the joint likelihood (1), the critical thing is to properly model \( P(d|q) \). We use the softmax function in the following form:
\[
P(d|q; \bar{w}) = \frac{\exp[(\gamma R(q,d;\bar{w})] + \sum_{d' \in D} \exp[(\gamma R(q,d';\bar{w})]}}{\sum_{d \in D} \exp[(\gamma R(q,d;\bar{w})]}
\]  
(2)
where \( \gamma > 0 \) is a smoothing factor that is empirically set and \( D^i \) is a set of \( J \geq 4 \) irrelevant documents under \( q \). \( R(q,d;\bar{w}) \) is a similarity function of \( q \) and \( d \), parametrized by \( \bar{w} \). One form of the similarity function is cosine distance, which can be expressed as
\[
R(q,d;\bar{w}) = \frac{\bar{x}^{(q)} \bar{x}^{(d)}}{\|\bar{x}^{(q)}\| \|\bar{x}^{(d)}\|},
\]  
(3)
where \( \bar{x}^{(q)} = \alpha(q;\bar{w}(q)) \) and \( \bar{x}^{(d)} = \beta(d;\bar{w}(d)) \) are the semantic vectors of \( q \) and \( d \), respectively, and \( \bar{w}(q) \) and \( \bar{w}(d) \) are parts of \( \bar{w} \) corresponding to \( q \) and \( d \), respectively.

By taking the negative logarithm of (1) we have the differentiable loss function as
\[
L(\bar{w}) = -\sum_i \ln P(d_i|q_i)
\]  
\[
= -\sum_i \ln \frac{\exp[(\gamma R(q_i,d_i;\bar{w})] + \sum_{d' \in D_i} \exp[(\gamma R(q_i,d';\bar{w})]}}{\sum_{d \in D} \exp[(\gamma R(q_i,d;\bar{w})]}
\]  
(4)
where \( D_i \) is a set of \( J \geq 4 \) irrelevant documents under \( q_i \). Therefore, the aim of the DSSM is to find the functions \( \alpha \) and \( \beta \) that are parameterized by \( \bar{w}(q) \) and \( \bar{w}(d) \) respectively such that the loss function (4) is minimized. Since the 1-th term can also be written as
\[
L_1(\bar{w}) = \ln \frac{1}{1 + \sum_{d \in D^i} \exp[-\gamma R(q_i,d;\bar{w})]}
\]  
(5)
where \( \Delta_i = R(q_i,d_i;\bar{w}) - R(q_i,d;\bar{w}) \), we can see that, intuitively, minimizing \( L(\bar{w}) \) is to jointly maximize the different between the relevant document and the irrelevant documents under each query.

In DSSM, the functions \( \alpha \) and \( \beta \) are represented by two neural networks. For both nets, we use tanh function as the activation function. That is if we denote the \( K \)-th layer as \( l^k_1, l^k_2, \ldots, l^k_n \) and the \( (k+1) \)-th layer as \( l^{k+1}_1, l^{k+1}_2, \ldots, l^{k+1}_{n+1} \), then for each \( i = 1, \ldots, n_{k+1} \),
\[
l^{k+1}_i = \frac{1 - \exp(-2a^k_i)}{1 + \exp(-2a^k_i)}
\]  
(6)
where \( a^k_i = \sum_{j=1}^{n_k} w^k_{ij} r^k_j + w^k_{0i} \). Note that the last layers for \( q \) and \( d \) respectively are \( \bar{x}^{(q)} \) and \( \bar{x}^{(d)} \). The structure is illustrated as the following Figure 1.

![Figure 1. Illustration of the DSSM. For both q and d, it maps the high dimensional sparse word-occurrence vector into low dimensional dense semantic vector.](image)

2.2. Text Representation
Since the dimension of the sparse word-occurrence vector representation of a text stream input can be very high because there can be unlimited words, we apply the letter-tri-gram (LTG) based text stream representation for the purpose of the dimension reduction. To illustrate the idea, consider the English text stream "Tom Cruise’s movies". We first add "#" at the head and the tail of the text stream and remove all the symbols other than the 26 English letters a-z and the 10 digits 0-9, we then change all the letters to lower case to obtain "#tomcruisemovies#". The idea of the LTG representation is to break the stream into (#to, tom, omc, cru, ru,uis, ises, esm, smo, mov, ovi, vie, ies, es9), which can be represented by a vector in a \( 36^2 \times 2 \times 36^2 = 49232 \)-dimensional space. The vector has 16 entries as 1 and all the other 49232 entries are 0. If the original word based dictionary has 500k unique words, then the LTG representation has a 10-fold reduction in dimensionality. Besides dimension reduction, another benefit of the LTG based representation is that the morphological variants of a same text stream can be mapped to close vectors in the LTG space. This is encouraging since a query can always have misspelled forms. Take an example #banana# vs. #bannana#, they are two misspelled forms of the word #banana# and have the same LTG words: #ba, ban, ana, nan, ann, nna, na#, with ana occurring twice.

Formally, if we denote the LTG dictionary as \( \{v_1, \ldots, v_K\} \), then a text stream can be represented as a \( K \)-dimensional vector \( \tilde{c} = (c_1, \ldots, c_K) \), where \( c_j \) is the count of the LTG word \( v_j \). And if only the 26 English
letters a-z and the 10 digits 0-9 are allowed, then \( K = 49248 \), which is the input dimension at the first layer.

2.3. Parameter Estimation

We apply the paralleled stochastic gradient descent method [12] to minimize the loss function (4). The one-step update can be expressed as

\[
\overline{w}^{(t+1)} = \overline{w}^{(t)} - \eta_t \nabla_w L(\overline{w})
\]  (7)

where \( \eta_t > 0 \) is the learning rate at \( t \)-th iteration. The gradient can be written as

\[
\nabla_w L(\overline{w}) = \sum_d \sum_{d_i \in D_d} \alpha_d \cdot \nabla_{\overline{w}} \Delta_d,
\]  (8)

where \( \alpha_d = \frac{-\gamma \exp(-\gamma \Delta_d)}{1 + \sum_{d_i \in D_d} \exp(-\gamma \Delta_d)} \). Hence computing the gradient of the loss function \( L(\overline{w}) \) is reduced to computing the gradient of the similarity function \( R(q, d; \overline{w}) \). It depends on the structures of the two neural nets that represent the two functions \( \alpha \) and \( \beta \), respectively. We adopt the same structure as the following Figure 2 shows.

![Figure 2. The structure of the neural network. The input layer corresponds to the LTG space that has dimension 49248, there are two intermediate layers of dimension 100, and the output layer corresponds to a semantic space that has dimension 50.](image)

2.4. Integration

The value of the DSSM is expected in two aspects: one is the document retrieval; the other is document ranking. Precisely we expect that by using the DSSM based similarity scores, previously missed relevant documents due to poor or none term match can be retrieved. We're also interested in seeing whether the ranker’s performance can be further lifted when the DSSM based similarity scores are added as the features in addition to the existing ones. Therefore, we integrate the DSSM in both the retrieval and the ranking. The following Figure 3 shows how the DSSM is used in our media search and ranking. Note that mapping a document \( d \) to its LTG-vector representation \( \tilde{x}^{(d)} \) can be performed offline. And since a query \( q \) usually contains much less terms, mapping it to its LTG-vector representation \( \tilde{x}^{(q)} \) can be performed in real-time [7][9].

3. Experiments

We trained three models in our experiments. First, we trained a baseline ranker that does not have the DSSM similarity scores as features. Second, we trained a DSSM and used it to compute \( R(q, d) \) for each \( (q, d) \)-pair in the ranker’s training data. Third, we added \( R(q, d) \) in different forms as similarity features to the existing features and trained a ranker that we call DSSM ranker. Since a media document usually contains title, description, actor list, genre, we calculate four similarity scores \( R_{\text{title}}, R_{\text{description}}, R_{\text{actors}}, \text{ and } R_{\text{genre}}. \) Consequently, we added four new similarity features to the existing feature set. Note that the new features should be added to both the training data set and the test data set.

3.1. DSSM Training

Training a DSSM needs huge amount of data because the size of the parameter space can be very large. As the Figure 2 shows, there are \( 49248 \times 100 + 100 \times 100 + 200 \times 100 + 100 \times 50 + 50 = 4,940,050 \) parameters in \( \overline{w}^{(q)} \) or \( \overline{w}^{(d)} \). However, there are only about 298,184 labeled query-document pairs with 17,735 unique queries in our training data for the Xbox’s ranker. These query-document pairs were from the Xbox’s click-logs from December 2013 to March 2014 and were labeled using the method in [11] that combines human judgments and click-signals. The label gains range from 0 to 15. We adopted the strategy of leveraging the Bing’s large web search logs and extracted from the year 2013’s logs about 20 million clicked query-document pairs as positive examples. By random reshuffling, we obtained additional 80 million query-document pairs that are treated as negative examples. In total, there are 100
million unclicked documents with one clicked document and four
unclicked documents under each query. We first trained
a DSSM using these 100 million pairs to obtain what we
call “seed model”, we then used the 298k labeled query-
document pairs in Xbox’s data to tune the seed model. In
the 298k labeled query-document pairs, we treat those
pairs with label gains no less than 7 as positive examples
and others as negative examples. Again by randomly
reshuffling the 298k query-document pairs, we obtained
additional 700k query-document pairs as negative
examples so that we have about 1 million pairs, with one
positive document and four negative documents under
each query, for the tuning.

3.2. Ranker Training

For the original training data of the xbox’s ranker that
contains 298k labeled query-document pairs, there are
about 10k existing features that are built from the basic
information on word matching, clicks, usages, release
date, and the advanced models including WTM, LSA,
PLSA, and LDA. To show the effect of the DSSM, we
trained a “control” ranker as the baseline with only the
existing features and a “treatment” ranker with existing
features plus the four new semantic similarity features
Rtitle, Rdescription, Ractors, and Rgenre. We used the gradient-
boosted decision trees [3][10] and the LambdaRank
algorithm [2] for learning the rankers. We used 1000
gradient boosted decision trees and for each individual
tree, we required that there are at least 200 data points in
a node.

3.3. Evaluation

Our evaluation approach is to test the DSSM in both
the retrieval and the ranking. We set up a baseline by
retrieving documents per query using the inverted-file-
index method and ranking the retrieved documents by
using the baseline ranker. To compare with the baseline,
we first changed the document retrieval method into the
criterion that the maximum DSSM similarity score is
positive, and we still used the baseline ranker to rank the
retrieved documents. We then change the baseline ranker
into the advanced one with the four new semantic
similarity features. For both the retrieval methods, we
retrieved documents for each sampled query-document pair,
we treat those pairs with label gains no less than 7 as positive examples
and others as negative examples. Again by randomly
reshuffling the 298k query-document pairs, we obtained
additional 700k query-document pairs as negative
examples so that we have about 1 million pairs, with one
positive document and four negative documents under
each query, for the tuning.

For both the two types of the labels, the mean
normalized discounted cumulative gain (NDCG) [8] can
be calculated. Specifically, we calculate the mean NDCG
at top i position as

\[
\text{NDCG}_i = \frac{1}{N} \sum_q \left( \sum_{j=1}^{i} \frac{s^q_j}{\log(2+j)} \right) / \left( \sum_{j=1}^{i} \frac{s^q_j}{\log(2+j)} \right).
\]

where \( s^q_1 \geq s^q_2 \geq \cdots \) represent the descending order
of \( s^q_1, s^q_2, \ldots \), which respectively are the label scores of
the documents at positions 1, 2, ... under the query \( q \) that
does not have all zero-score results, and \( N \) is total
number of such queries.

3.4. Results

The main experimental results are summarized in
Table 1 and Table 2. The Table 1 corresponds to the evaluation
data with human labels; the Table 2
corresponds to the evaluation data with labels as numbers
of users who clicked. The mean NDCG numbers are all
relative ones. For each \( i = 1, 4, 8 \), the mean NDCG value
of basic retrieval + baseline ranker is scaled to 1, the
mean NDCG values of DSSM retrieval + baseline ranker
and DSSM retrieval + DSSM ranker are multipliers. It
can be seen that DSSM improves the results of both the
retrieval and the ranking under the two NDCG metrics.
We also performed significant tests across all queries
using the paired t-test and the two-sample t-test. All the
p-values are far less than 0.01.

![Table 1. NDCG values by human labels](image)

<table>
<thead>
<tr>
<th>Method</th>
<th>NDCG_1</th>
<th>NDCG_4</th>
<th>NDCG_8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic retrieval + Baseline ranker</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>DSSM retrieval + Baseline ranker</td>
<td>1.0200</td>
<td>1.0143</td>
<td>1.0121</td>
</tr>
<tr>
<td>DSSM retrieval + DSSM ranker</td>
<td>1.0438</td>
<td>1.0414</td>
<td>1.0289</td>
</tr>
</tbody>
</table>

Number of queries: 7029
Number of query-document pairs: 37,948
Table 2. NDCG values by number of users who clicked

<table>
<thead>
<tr>
<th></th>
<th>NDCG&lt;sub&gt;1&lt;/sub&gt;</th>
<th>NDCG&lt;sub&gt;4&lt;/sub&gt;</th>
<th>NDCG&lt;sub&gt;8&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic retrieval + Baseline ranker</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>DSSM retrieval + Baseline ranker</td>
<td>1.0186</td>
<td>1.0095</td>
<td>1.0121</td>
</tr>
<tr>
<td>DSSM retrieval + DSSM ranker</td>
<td>1.0361</td>
<td>1.0235</td>
<td>1.0329</td>
</tr>
</tbody>
</table>

Number of queries: 22828
Number of query-document pairs: 181692

To illustrate by example, the following Table 3 lists the top 10 results for the query “arnold schwarzenegger” under the two settings: basic retrieval + baseline ranker vs. DSSM retrieval + DSSM ranker. Under the setting basic retrieval + baseline ranker, the new hot movie “Sabotage” stared by Arnold Schwarzenegger is not successfully retrieved. On the other hand, the DSSM can capture the relation between “arnold schwarzenegger” and the character name “John Breacher Wharton” and successfully retrieve the movie. Moreover, the movie is also ranked the highest by the advanced ranker. Another movie “Generation Iron” that is a documentary on top bodybuilders in sport is also not successfully retrieved under the setting basic retrieval + baseline ranker. Again, the DSSM can capture the connection between Arnold Schwarzenegger and the bodybuilding and, therefore, retrieve the movie. This movie is also ranked high.

For the very old movie “Conan the Barbarian”, although it’s a relevant result, compared with other results on the list, it’s much less satisfactory. The DSSM features, which have the relevance label gain information incorporated, have a certain penalty on this movie so that it’s excluded from the top 10 results under the setting DSSM retrieval + DSSM ranker.

Table 3. A sample comparison of two settings

Query = “arnold schwarzenegger”

<table>
<thead>
<tr>
<th></th>
<th>Basic retrieval + Baseline ranker</th>
<th>DSSM retrieval + DSSM ranker</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The Last Stand, movie, 2013</td>
<td>Sabotage, movie, 2014</td>
</tr>
<tr>
<td>3</td>
<td>Total Recall, movie, 1990</td>
<td>The Last Stand, move, 2013</td>
</tr>
</tbody>
</table>

4. Conclusions

In this paper, we presented some results of applying the DSSM model to build the similarity features for document retrieval and ranking in media search. Theoretically, the advantage of DSSM lies in that it captures the relatedness between a query and a document at the semantic level other than term level. Therefore, for a query, a relevant document with poor or even no term match can still be retrieved. In addition, the DSSM model entertains the supervise learning so that the label information can be reflected in the similarity scores. To show the benefits of the DSSM similarity features, we set up a baseline that uses the inverted-file-index method for document retrieval and a ranker with only existing features for ranking the retrieved documents. We first changed the basic retrieval method into the DSSM similarity score and then changed the basic ranker into the advanced one with the DSSM similarity scores as new features. We obtained a DSSM by first training the seed model from the large search logs and then tuning the model by using the ranker’s training data. We used two evaluation metrics. One is the mean NDCG value from human labels, the other is the mean NDCG value from a number of users who clicked. The first measures relevance; the second measures user engagement. Our experiments have shown that the DSSM improves both the document retrieval and the ranking results.

Acknowledgments

We thank Xinying Song for building the DSSM training and evaluation pipeline. We thank the Microsoft
Research Group for providing the GPU computing environment for training and tuning the DSSM. We also thank the Microsoft Aether team for providing the computing environment for training and evaluating rankers.

References


