Learning Deep Structured Semantic Models for Web Search using Clickthrough Data

Po-Sen Huang¹, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, Larry Heck
Microsoft Research, Redmond, WA, USA

Presented at CIKM, Oct. 2013

¹P. Huang is with UIUC. He was an intern with MSR when this work was done.
Background of Web Search

• Traditionally, search engines retrieve web documents by matching terms in documents with those in a search query – **lexical matching**
• However, lexical matching can be suboptimal due to language discrepancy between documents and queries
  • E.g., a concept can often be expressed using different vocabularies and language styles
• Need to bridge the lexical gaps between queries and documents – **semantic matching**
Related work on semantic modeling for IR

• Document retrieval based on semantic content
  • Deal with lexicon mismatch between search queries and web documents

• Early approaches
  • Latent Semantic Analysis (LSA) and its varieties (Deerwester et al., 1990)
    • LSA extracts abstract semantic content using SVD
  • Many extensions exist: PLSA, LDA, etc.

• Recent improvements:
  • Go deeper: e.g., semantic hashing (Hinton and Salakhutdinov 2011)
  • Go beyond documents: e.g., using click signals (Gao et al. 2010; Gao et al. 2011)
Previous work: Clickthrough Log based models

- State of the art document ranking approaches that use models trained on clickthrough data.
  - Oriented PCA (Diamantaras et al., 1996)
  - Word Translation Model (Gao et al. 2010)
  - Bilingual Topic Model (Gao et al. 2011)
  - Discriminative Projection Model (Yih et al. 2011; Gao et al. 2011)

- However,
  - Expressive power could be limited by using linear model
  - Not scalable, model size increases rapidly along vocabulary size
Previous work: Deep auto encoder

• Training
  • Step1: RBM layer-wise pre-training, initialize weights
  • Step2: Deep auto-encoder, learn internal representations through minimizing reconstruction error

(Hinton and Salakhutdinov 2011)
Previous work: Deep auto encoder (II)

• Testing
  • Project both query and document to a common semantic space
  • Measure the relevance of Q and D in that space directly

\[
Relevance(Q, D) = \text{Cosine}(y_Q, y_D)
\]

Semantic space

Multiple layers of non-linear projections

Raw term vector

Query

Document
Problems of DAE

- Mismatched *learning objective*
  - Model is trained by reconstructing the document, not for relevance measure
- Lack of *scalability*
  - Model size increases rapidly along the vocabulary size

\[ 5000 \quad 2000 \quad 128 \quad 500 \quad 500K \]

Model size in previous studies (Hinton 2011)

\[ 128 \quad 500 \quad 500 \quad 2000 \]

~1 million parameters

\[ 128 \quad 500 \quad 500 \quad 500K \]

Model size of a (small) real-world Web search task

250 million parameters
Learning semantic representations from Web and search logs

- The goal of deep semantic representation for web search
  - Map docs/queries/entities/... to a common semantic space for inference

- **Our solution**: Deep Structured Semantic Models (DSSM)
  - Using the *tri-letter* based word hashing for scalable word representation
  - Using the *deep neural net* to extract high-level semantic representations
  - Using the *click signal* to guide the learning
Tri-letter: a scale-able word representation

• Tri-letter based Word Hashing of "cat"
  • \( \rightarrow \#cat\# \)
  • Tri-letters: \#-c-a, c-a-t, a-t-\#.

• Compact representation
  • \(|\text{Voc}| (500K) \rightarrow |\text{TriLetter}| (30K)\)
  • Generalize to unseen words
  • Robust to misspelling, inflection, etc.

\[
x (\text{cat}) = \begin{bmatrix}
0 \\
\vdots \\
1 \\
\vdots \\
0
\end{bmatrix}
\]

The index of word \textit{cat} in the vocabulary

\[
f (\text{cat}) = \begin{bmatrix}
0 \\
\vdots \\
1 \\
\vdots \\
1 \\
\vdots \\
0
\end{bmatrix}
\]

Indices of \#-c-a, c-a-t, a-t-\# in the letter-tri-gram list, respectively.
Word hashing by n-gram of letters

- Collision:
  - What if different words have the same word hashing vector?
  - Statistics
    - 22 out of 500K words collide
    - Collision Example: #bananna# <-> #bannana#

<table>
<thead>
<tr>
<th>Vocabulary size</th>
<th>Unique tri-letter observed in voc</th>
<th>Number of Collisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>40K</td>
<td>10306</td>
<td>2</td>
</tr>
<tr>
<td>500K</td>
<td>30621</td>
<td>22</td>
</tr>
</tbody>
</table>
Use deep neural nets for semantic representation extraction

Use tri-letter based word hashing to handle any unseen words

Maximize the cosine similarity between the query and the clicked doc

Training DSSM

- Optimization: SGD (w/ minibatch)
- Objective: Cosine loss defined on the clickthrough data
  - For each query $Q$, there is a set of documents $D$
    - $D = \{D^+, D^-_1, ..., D^-_N\}$ includes the clicked doc $D^+$, and a set of unclicked docs collected via sampling
  - $R(Q, D) = \text{Cosine}(y_D, y_Q)$
  - $P(D | Q) = \frac{\exp(\gamma R(Q, D))}{\sum_{D', \in D} \exp(\gamma R(Q, D'))}$
  - $\text{loss}(Q, D) = -\log P(D^+ | Q)$
Implementation Details

- Select parameters based on cross validation
- Randomly choose 4 competitors (similar performance as selecting based on TF-IDF ranking)
- We fixed the architecture to be
  - TriLetter-300-300-128
- Tanh() as the activation function
- Random initialization – pretraining does not make much difference
- Use stochastic gradient descent to optimize the training objective
- Control learning rate
### NDCG results on a real-world Web search task

<table>
<thead>
<tr>
<th>Models</th>
<th>NDCG@1</th>
<th>NDCG@3</th>
<th>NDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM25</td>
<td>30.8</td>
<td>37.3</td>
<td>45.5</td>
</tr>
<tr>
<td>Previous Shallow/Deep Semantic Models, trained on doc collection (unsupervised)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSA (Deerwester et al., 1990)</td>
<td>29.8</td>
<td>37.2</td>
<td>45.5</td>
</tr>
<tr>
<td>PLSA (Hofmann 1999)</td>
<td>29.5</td>
<td>37.1</td>
<td>45.6</td>
</tr>
<tr>
<td>Deep Auto-Encoder (Hinton et al., 2011)</td>
<td>30.6</td>
<td>37.4</td>
<td>45.6</td>
</tr>
<tr>
<td>Previous Semantic Models trained on click logs (supervised)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPM (w/ S2Net (Yih et al., 2011))</td>
<td>32.9</td>
<td>40.1</td>
<td>47.9</td>
</tr>
<tr>
<td>Word Translation Model (Gao et al, 2010)</td>
<td>33.2</td>
<td>40.0</td>
<td>47.8</td>
</tr>
<tr>
<td>Bilingual Topic Model (Gao et al., 2011)</td>
<td>33.7</td>
<td>40.3</td>
<td>48.0</td>
</tr>
<tr>
<td>Our deep structured semantic model trained on click logs (supervised)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSSM (this work)</td>
<td><strong>36.2</strong></td>
<td>42.5</td>
<td>49.8</td>
</tr>
</tbody>
</table>

(Please refer to our CIKM13 paper for more details)
Visualization

\[ \hat{x} = \text{argmax}_x (h(x)) \]

<table>
<thead>
<tr>
<th>Car</th>
<th>Holiday</th>
<th>Video</th>
<th>Hunting</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>automotive wheels</td>
<td>happy</td>
<td>youtube</td>
<td>bear</td>
<td>systems</td>
</tr>
<tr>
<td>cars</td>
<td>lyrics</td>
<td>videos</td>
<td>hunting</td>
<td>protect</td>
</tr>
<tr>
<td>auto</td>
<td>musical</td>
<td>dvd</td>
<td>texas</td>
<td>platform</td>
</tr>
<tr>
<td>car</td>
<td>halloween</td>
<td>downloads</td>
<td>colorado</td>
<td>efficiency</td>
</tr>
<tr>
<td>vehicle</td>
<td>eastern</td>
<td>movie</td>
<td>hunter</td>
<td>oems</td>
</tr>
<tr>
<td></td>
<td>festival</td>
<td>cd</td>
<td>tucson</td>
<td>systems32</td>
</tr>
</tbody>
</table>

**Table 1:** Examples of words with high activation at the same nodes.
Summary

• Proposed a deep structured semantic model (DSSM) for web search
  
  **Tri-letter** based word representation

  **deep neural net** based semantic model

  **Cosine-similarity based loss function** defined on click log

• Significant gains over previous approaches
  
  • 5 pt NDCG gain compared with BM25
  
  • 3 pt gain compare with state of the art latent semantic models (BLTM, MT, DPM, etc.)


Word hashing by n-gram of letters

- Collision:
  - What if different words have the same word hashing vector?
  - Statistics
    - 22 out of 500K words collide
    - Collision Example: #banana# <-> #bannana#

<table>
<thead>
<tr>
<th>Vocabulary</th>
<th>Type</th>
<th>Unique Key</th>
<th>Collision</th>
</tr>
</thead>
<tbody>
<tr>
<td>40K</td>
<td>Bigram</td>
<td>1107</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Trigram</td>
<td>10306</td>
<td>2</td>
</tr>
<tr>
<td>500K</td>
<td>Bigram</td>
<td>1607</td>
<td>1192</td>
</tr>
<tr>
<td></td>
<td>Trigram</td>
<td>30621</td>
<td>22</td>
</tr>
</tbody>
</table>