Semantic Matching in Search

Jun Xu
junxu@ict.ac.cn

Institute of Computing Technology
Chinese Academy of Sciences
People Who Also Contributed to This Tutorial

Hang Li
Outline of Tutorial

• Semantic Matching between Query and Document

• Approaches to Semantic Matching
  1. Matching by Query Reformulation
  2. Matching with Term Dependency Model
  3. Matching with Translation Model
  4. Matching with Topic Model
  5. Matching with Latent Space Model

• Summary
A Good Web Search Engine

• Must be good at
  – Relevance
  – Coverage
  – Freshness
  – Response time
  – User interface

• Relevance is particularly important
Query Document Mismatch Challenge

Table 1.1: Examples of query document mismatch.

<table>
<thead>
<tr>
<th>query</th>
<th>document</th>
<th>term match</th>
<th>semantic match</th>
</tr>
</thead>
<tbody>
<tr>
<td>seattle best hotel</td>
<td>seattle best hotels</td>
<td>partial</td>
<td>yes</td>
</tr>
<tr>
<td>pool schedule</td>
<td>swimming pool schedule</td>
<td>partial</td>
<td>yes</td>
</tr>
<tr>
<td>natural logarithm transform</td>
<td>logarithm transform</td>
<td>partial</td>
<td>yes</td>
</tr>
<tr>
<td>china kong</td>
<td>china hong kong</td>
<td>partial</td>
<td>no</td>
</tr>
<tr>
<td>why are windows so expensive</td>
<td>why are macs so expensive</td>
<td>partial</td>
<td>no</td>
</tr>
</tbody>
</table>
Why Query Document Mismatch Happens?

• Search is still mainly based on term level matching
• Same intent can be represented by different queries (representations)
• Query document mismatch occurs, when searcher and author use different terms (representations) to describe the same concept
Same Search Intent
Different Query Representations

<table>
<thead>
<tr>
<th>Query 1</th>
<th>Query 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>“how far” earth sun</td>
<td>average distance from the earth to the sun</td>
</tr>
<tr>
<td>“how far” sun</td>
<td>how far away is the sun from earth</td>
</tr>
<tr>
<td>average distance earth sun</td>
<td>average distance from earth to sun</td>
</tr>
<tr>
<td>how far from earth to sun</td>
<td>distance from earth to the sun</td>
</tr>
<tr>
<td>distance from sun to earth</td>
<td>distance between earth and the sun</td>
</tr>
<tr>
<td>distance between earth &amp; sun</td>
<td>distance from the earth to the sun</td>
</tr>
<tr>
<td>how far earth is from the sun</td>
<td>distance from the earth to the sun</td>
</tr>
<tr>
<td>distance between earth sun</td>
<td>distance from the sun to the earth</td>
</tr>
<tr>
<td>distance of earth from sun</td>
<td>distance from the sun to earth</td>
</tr>
<tr>
<td>“how far” sun earth</td>
<td>how far away is the sun from the earth</td>
</tr>
<tr>
<td>how far earth from sun</td>
<td>distance between sun and earth</td>
</tr>
<tr>
<td>how far from earth is the sun</td>
<td>how far from the earth to the sun</td>
</tr>
<tr>
<td>distance from sun to the earth</td>
<td></td>
</tr>
</tbody>
</table>

Table 1.2: Queries about “distance between sun and earth”.
### Same Search Intent Different Query Representations

#### Table 1.3: Queries about “Youtube”.

<table>
<thead>
<tr>
<th>youtub</th>
<th>youtubr</th>
<th>youtubcom</th>
<th>youtubcomyourtube</th>
<th>you</th>
<th>yourtube</th>
<th>www you tube</th>
<th>utube</th>
<th>yourtube com</th>
<th>www utube</th>
<th>utube com</th>
<th>utub</th>
<th>my tube</th>
<th>our tube</th>
</tr>
</thead>
<tbody>
<tr>
<td>youtube</td>
<td>youtube</td>
<td>youtube com</td>
<td>youtube music videos</td>
<td>youtub</td>
<td>youtub com</td>
<td>www youtube</td>
<td>youtube videos</td>
<td>youtubeco</td>
<td>www youtube co</td>
<td>www utubecom</td>
<td>utube</td>
<td>utube videos</td>
<td>toutube</td>
</tr>
</tbody>
</table>
• Reason for mismatch: language understanding by computer is hard, if not impossible
• A more realistic approach: avoid understanding and conduct *matching*
Aspects of Semantic Matching

• More aspects of the query and document can match, more likely the query and document are relevant
  – **Form**: onecar → onecare
  – **Phrase**: “hot dog” → “hot dog”
  – **Sense**: NY → New York
  – **Topic**: Microsoft Office → Microsoft, PowerPoint, Word, Excel...
  – **Structure**: how far is sun from earth → distance between sun and earth
Semantic Matching in Search

User Interface → Query Understanding → Retrieving

User Interface → Ranking → Query Document Matching

User Interface → Crawling → Document Understanding → Indexing

Web → Crawling → Document Understanding → Indexing
Query Understanding

Structure Identification

Topic Identification

Similar Query Finding

Phrase Identification

Spelling Error Correction

main phrase: michael jordan

structure

topic: machine learning, berkeley

sense

similar query: michael i. jordan

phrase: michael jordan

phrase: berkeley

phrase

query form: michael jordan berkeley

term

michael jordan berkele
Document Understanding

- **Title Structure Identification**
  - main phrase in title: michael jordan
  - Structure

- **Topic Identification**
  - topic: machine learning, berkeley
  - Topic

- **Key Phrase Identification**
  - key phrase: michael jordan, professor, electrical engineering
  - Key Phrase

- **Phrase Identification**
  - phrase: michael jordan, professor, department, electrical engineering
  - Phrase

Homepage of Michael Jordan

Michael Jordan is Professor in the Department of Electrical Engineering

......
Query Document Matching

Query form: michael jordan berkeley
Similar query: michael i jordan
Main phrase: michael jordan
Phrase: michael jordan, berkeley
Topic: machine learning

Document: michael jordan homepage
Main phrase in title: michael jordan
Key phrase: michael jordan, berkeley
Phrase: michael jordan, professor, department of electrical engineering
Topic: machine learning, berkeley
Semantic Matching and Semantic Search

Semantic Search

Query: unstructured data

Documents: unstructured data

Knowledge base: structured data

Semantic Matching

Query: unstructured data

Documents: unstructured data
Matching and Ranking

- In search, first matching and then ranking
- Matching results as features for ranking

<table>
<thead>
<tr>
<th></th>
<th>Matching</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction</td>
<td>Matching degree between one query and one document</td>
<td>Ranking a list of documents</td>
</tr>
<tr>
<td>Model</td>
<td>$f(q, d)$</td>
<td>$f(q, {d_1, d_2, \ldots, d_N})$</td>
</tr>
<tr>
<td>Challenge</td>
<td>Mismatch</td>
<td>Correct ranking on the top</td>
</tr>
</tbody>
</table>
## Semantic Matching in Other Tasks

<table>
<thead>
<tr>
<th>task</th>
<th>types of texts</th>
<th>relation between texts</th>
</tr>
</thead>
<tbody>
<tr>
<td>search</td>
<td>A=query, B=document</td>
<td>relevance</td>
</tr>
<tr>
<td>question answering</td>
<td>A=question, B=answer</td>
<td>answer to question</td>
</tr>
<tr>
<td>cross-language IR</td>
<td>A=query, B=document (in diff. lang.)</td>
<td>relevance</td>
</tr>
<tr>
<td>short text conversation</td>
<td>A=text, B=text</td>
<td>message and comment</td>
</tr>
<tr>
<td>similar document detection</td>
<td>A=text, B=text</td>
<td>similarity</td>
</tr>
<tr>
<td>online advertising</td>
<td>A=query, B=ads.</td>
<td>relevance as ads.</td>
</tr>
<tr>
<td>paraphrasing</td>
<td>A=sentence, B=sentence</td>
<td>equivalence</td>
</tr>
<tr>
<td>textual entailment</td>
<td>A=sentence, B=sentence</td>
<td>entailment</td>
</tr>
</tbody>
</table>
Learning to Match

\[
\begin{align*}
    X & \quad \quad \quad \quad \quad \quad \quad Y \\
    x_1 & \quad r_1 & \quad y_1 \\
    x_2 & \quad r_2 & \quad y_2 \\
    x_N & \quad r_N & \quad y_N
\end{align*}
\]

\[
\begin{align*}
    X & \quad \quad \quad \quad \quad \quad \quad Y \\
    x & \quad ? & \quad y
\end{align*}
\]

Learning System

\[\arg \min_{f \in F} \sum_{i=1}^{N} L(r_i, f(x_i, y_i)) + \Omega(f)\]

Model \(f(x, y)\)

Matching System

\[f(x, y)\]
Challenges

• How to leverage relations in data and prior knowledge
• How to scale up
• How to deal with tail
Approaches to Semantic Matching Between Query and Document

• Matching by Query Reformulation
• Matching with Term Dependency Model
• Matching with Translation Model
• Matching with Topic Model
• Matching with Latent Space Model
Outline of Tutorial

• Semantic Matching between Query and Document

• Approaches to Semantic Matching
  1. Matching by Query Reformulation
  2. Matching with Term Dependency Model
  3. Matching with Translation Model
  4. Matching with Topic Model
  5. Matching with Latent Space Model

• Summary
Query Reformulation

• Transforming the original query to queries (representations) that can better match with documents in the sense of relevance

• Also called
  – Query transformation
  – Query re-writing
  – Query refinement
  – Query alternation
Query Transformation

- Our focus is on how queries can be transformed to equivalent, potentially better, queries
  - Queries into paraphrases or “translations”
  - Long queries into shorter queries
  - Short queries into longer queries
  - Queries in one domain to queries in other domains
  - Unstructured queries into structured queries

From Bruce Croft, ECIR 2009
### Types of Query Reformulation

<table>
<thead>
<tr>
<th>type</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>spelling error correction</td>
<td>mlss singapore → miss singapore</td>
</tr>
<tr>
<td>merging</td>
<td>face book → facebook</td>
</tr>
<tr>
<td>splitting</td>
<td>dataset → data set</td>
</tr>
<tr>
<td>stemming</td>
<td>seattle best hotel → seattle best hotels</td>
</tr>
<tr>
<td>synonym</td>
<td>ny times → new york times</td>
</tr>
<tr>
<td>segmentation</td>
<td>new work times square → “new york” “times square”</td>
</tr>
<tr>
<td>query expansion</td>
<td>www → www conference</td>
</tr>
<tr>
<td>query deduction</td>
<td>natural logarithm transformation → logarithm transformation</td>
</tr>
<tr>
<td>stopword removal\preservation</td>
<td>the new year → “the new year”</td>
</tr>
<tr>
<td>paraphrasing</td>
<td>how far is sun from earth → distance between sun and earth</td>
</tr>
</tbody>
</table>

1 “The new year” is the title of an American movie, and thus the word “the” should not be removed here, although it is usually treated as stopword.
Problems in Query Reformulation

- Query Reformulation
- Similar Query Mining
- Blending
Query Reformulation Problem

• Task
  – Rewrite original query to (multiple) similar queries

• Challenge
  – Topic drift

• Current situation
  – In practice, mainly limited to spelling error correction, query segmentation etc.
Query Reformulation is Difficult

• Depending on the contents of both query and document

• Except
  – Spelling error correction
  – Definite splitting and merging: face book → facebook
  – Definite segmentation: “hot dog”
Methods of Query Reformulation

• Generative approach
  – Source channel model (Brill & Moore, ’00; Cucerzan & Brill, ’04; Duan & Hsu, ‘10)

• Discriminative approach
  – Max entropy (Li et al., ‘06)
  – Log linear model (Okazaki et al., ’08; Wang et al., ‘11)
  – Conditional Random Fields (Guo et al., ‘08)
Conditional Random Field for Query Reformulation (Guo et al., ‘08)

- \(x\): observed noisy query, e.g., window onecar
- \(y\): reformulated query, e.g., windows onecare
- \(o\): a sequence of operations
- Learning: \(P(y, o|x)\)
- Prediction: \(\text{argmax}_{y,o} P(y, o|x)\)

\[
\Pr(y, o|x) = \frac{1}{Z(x)} \prod_{i=1}^{n} \phi(y_{i-1}, y_i) \phi(y_i, o_i, x)
\]

![Diagram of CRF and CRF-QR models]
## Operations

<table>
<thead>
<tr>
<th>Task</th>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spelling Correction</td>
<td>Deletion</td>
<td>Delete a letter in the word</td>
</tr>
<tr>
<td></td>
<td>Insertion</td>
<td>Insert a letter into the word</td>
</tr>
<tr>
<td></td>
<td>Substitution</td>
<td>Replace a letter in the word with another letter</td>
</tr>
<tr>
<td></td>
<td>Transposition</td>
<td>Switch two letters in the word</td>
</tr>
<tr>
<td>Word Splitting</td>
<td>Splitting</td>
<td>Split one word into two words</td>
</tr>
<tr>
<td>Word Merging</td>
<td>Merging</td>
<td>Merge two words into one word</td>
</tr>
<tr>
<td>Phrase Segmentation</td>
<td>Begin</td>
<td>Mark a word as beginning of phrase</td>
</tr>
<tr>
<td></td>
<td>Middle</td>
<td>Mark a word as middle of phrase</td>
</tr>
<tr>
<td></td>
<td>End</td>
<td>Mark a word as end of phrase</td>
</tr>
<tr>
<td></td>
<td>Out</td>
<td>Mark a word as out of phrase</td>
</tr>
<tr>
<td>Word Stemming</td>
<td>+s/-s</td>
<td>Add or Remove suffix ‘-s’</td>
</tr>
<tr>
<td></td>
<td>+ed/-ed</td>
<td>Add or Remove suffix ‘-ed’</td>
</tr>
<tr>
<td></td>
<td>+ing/-ing</td>
<td>Add or Remove suffix ‘-ing’</td>
</tr>
<tr>
<td>Acronym Expansion</td>
<td>Expansion</td>
<td>Expand acronym</td>
</tr>
</tbody>
</table>
Extended Model

\[
\Pr(y, \bar{\sigma}, \bar{z} | x) = \frac{1}{Z(x)} \prod_{i=1}^{n} (\phi(y_{i-1}, y_i)) \prod_{j_i=1}^{m_i} \phi(z_{i,j_i} | o_{i,j_i}, z_{i,j_i-1})
\]
Experimental Results

- Data: 10,000 queries, 6,421 queries were refined by human annotators
- Result: extended CRF-QR model significantly outperformed the baselines

<table>
<thead>
<tr>
<th></th>
<th>Pre.</th>
<th>Rec.</th>
<th>F1</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRF-QR</td>
<td>54.48</td>
<td>40.75</td>
<td>46.63</td>
<td>56.27</td>
</tr>
<tr>
<td>Cascaded1</td>
<td>53.38</td>
<td>39.71</td>
<td>45.54</td>
<td>55.57</td>
</tr>
<tr>
<td>Cascaded2</td>
<td>53.38</td>
<td>39.71</td>
<td>45.54</td>
<td>55.57</td>
</tr>
<tr>
<td>Cascaded3</td>
<td>53.38</td>
<td>39.71</td>
<td>45.54</td>
<td>55.57</td>
</tr>
<tr>
<td>Cascaded4</td>
<td>53.45</td>
<td>39.76</td>
<td>45.60</td>
<td>55.60</td>
</tr>
<tr>
<td>Cascaded5</td>
<td>53.45</td>
<td>39.76</td>
<td>45.60</td>
<td>55.60</td>
</tr>
<tr>
<td>Cascaded6</td>
<td>53.45</td>
<td>39.76</td>
<td>45.60</td>
<td>55.60</td>
</tr>
<tr>
<td>Generative</td>
<td>30.46</td>
<td>32.95</td>
<td>31.66</td>
<td>39.10</td>
</tr>
</tbody>
</table>
Similar Query Mining

• Task
  – Given click-through data for search session data
  – Find similar queries or similar query patterns
    E.g., ny → new York; distance tween X and Y → how far is X from Y

• Challenge
  – Dealing with noise
Mining of Similar Queries

Click-through data

q1 \rightarrow d1
q2 \rightarrow d2
\vdots
qm \rightarrow dn

Similar queries can be found by co-click

Search session data

\begin{itemize}
  \item q1
  \item q’
\end{itemize}

\begin{itemize}
  \item qn
  \item q’
\end{itemize}

Similar queries can be found from users’ query reformulations
Methods of Similar Query Mining

• Using click-through data
  – Pearson correlation coefficient (Xu & Xu, ’11)
  – Agglomerative clustering (Beeferman & Burger, ’00),
    DBScan (Wen et al., ’01), K-means (Baeza-Yates et al., ‘04),
    Query stream clustering (Cao et al., ’08; Liao et al., ‘12)
• Using search session data
  – Jaccard similarity (Huang et al., ’03), likelihood ratio (Jones
    et al., ‘06)
• Learning of query reformulation patterns
  – Mining natural language question patterns (Xue et al., ‘12)
• Learning of query similarity
  – Query similarity as metric learning (Xu & Xu ‘11)
Query Similarity as Metric Learning (Xu & Xu, ’11)

• Given similar query pairs and dissimilar query pairs
• Learn from head queries and propagate to tail queries
• Objective function:

\[
\max_{M \geq 0} \sum_{(q_i, q_j) \in S_+} \frac{\phi(q_i)^T M \phi(q_j)}{\sqrt{\phi(q_i)^T M \phi(q_i)} \sqrt{\phi(q_j)^T M \phi(q_j)}} \\
- \sum_{(q_i, q_j) \in S_-} \frac{\phi(q_i)^T M \phi(q_j)}{\sqrt{\phi(q_i)^T M \phi(q_i)} \sqrt{\phi(q_j)^T M \phi(q_j)}} - \lambda \| M \|_1
\]
Query Similarity as Metric Learning

- $\phi(q)$: N-gram vector space

<table>
<thead>
<tr>
<th>Query</th>
<th>Vectors in n-gram vector space</th>
</tr>
</thead>
<tbody>
<tr>
<td>NY times</td>
<td>(1, 0, 0, 1, 1, 0, ... )</td>
</tr>
<tr>
<td>New York times times</td>
<td>(0, 1, 1, 1, 0, 1, ... )</td>
</tr>
</tbody>
</table>

- Learned similarity function ($M$ is positive semi-definite)

$$sim(\phi(q_i), \phi(q_i)) = \frac{\phi(q_i)^T M \phi(q_j)}{\sqrt{\phi(q_i)^T M \phi(q_i)} \sqrt{\phi(q_j)^T M \phi(q_j)}}$$
Query Similarity as Metric Learning

• Interpretation: transformation between n-gram spaces
Experimental Results

- Constantly outperforms the two baselines on rare queries

Precision of similar query calculation methods on rare query

- P@1
- P@2
- P@3
Blending Problem

• Steps
  – Rewrite original query to multiple similar queries
  – Retrieve with multiple queries
  – Blend results from multiple queries

• Challenges
  – System to sustain searches with multiple queries
  – Blending model: matching scores are not comparable across queries
Blending

Input query
Michael Jordan

Similar queries
Michael I. Jordan
Michael Jordan NBA
Michael Jordan Berkeley

Retrieved documents

Re-ranking
Methods of Blending

• Linear combination (Xue et al., ‘08)
• Learning to rank (Sheldon et al., ‘11)
• Kernel methods (Wu et al., ‘11)
Kernel Method for Blending (Wu et al., ’11)

• Given query similarity and document similarity

• “Smoothing query and document similarity” by those of similar queries and documents

• Interpretation: nearest neighbor in space of query and document pair (double KNN)

• Automatically learning the weights of combination from click-data
Learning of Matching Model

- Matching function: $k(x, y) = \langle \phi_X(x), \phi_Y(y) \rangle_H$
- Input: training data $S = \{(x_i, y_i), r_i\}_{1 \leq i \leq N}$
- Output: matching function
- Optimization

$$\min_{k \in \mathcal{K}} \frac{1}{N} \sum_{i=1}^{N} l(k(x_i, y_i), r_i) + \Omega(k)$$
Learning of Matching Model Using Kernel Method

• Assumption: space of matching functions is RKHS generated by positive definite kernel $\tilde{k}: (X \times X \times X \times X) \rightarrow \mathbb{R}$.
Kernel Method

Query-document pair space

Query space

Hilbert space $k_Q(q, q')$

Hilbert space $k_{IR}(q, d)$

Hilbert space $k_{IR}(q', d')$

Matching

Hilbert space $k_D(d, d')$

Document space

Similarity Functions
Implementation: Learning of BM25

- BM25: similarity function between query and document, denoted as $k_{BM25}$
- Kernel:
  \[
  \overline{k}\left((q, d), (q', d')\right) = k_{BM25}(q, d)k_Q(q, q')k_D(d, d')k_{BM25}(q', d')
  \]
- Solution (called Robust BM25)
  \[
  k_{RBM25} = k_{BM25}(q, d) \sum_{i=1}^{N} \alpha_i k_Q(q, q_i)k_D(d, d_i)k_{BM25}(q_i, d_i)
  \]
### Experimental Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MAP</th>
<th>NDCG@1</th>
<th>NDCG@3</th>
<th>NDCG@5</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Web search</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robust BM25</td>
<td>0.1192</td>
<td>0.2480</td>
<td>0.2587</td>
<td>0.2716</td>
</tr>
<tr>
<td>Pairwise Kernel</td>
<td>0.1123</td>
<td>0.2241</td>
<td>0.2418</td>
<td>0.2560</td>
</tr>
<tr>
<td>Query Expansion</td>
<td>0.0963</td>
<td>0.1797</td>
<td>0.2061</td>
<td>0.2237</td>
</tr>
<tr>
<td>BM25</td>
<td>0.0908</td>
<td>0.1728</td>
<td>0.2019</td>
<td>0.2180</td>
</tr>
<tr>
<td><strong>Enterprise search</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robust BM25</td>
<td>0.3122</td>
<td>0.4780</td>
<td>0.5065</td>
<td>0.5295</td>
</tr>
<tr>
<td>Pairwise Kernel</td>
<td>0.2766</td>
<td>0.4465</td>
<td>0.4769</td>
<td>0.4971</td>
</tr>
<tr>
<td>Query Expansion</td>
<td>0.2755</td>
<td>0.4076</td>
<td>0.4712</td>
<td>0.4958</td>
</tr>
<tr>
<td>BM25</td>
<td>0.2745</td>
<td>0.4246</td>
<td>0.4531</td>
<td>0.4741</td>
</tr>
</tbody>
</table>

- Robust BM25 significantly outperforms the baselines, in terms of all measures on both data sets
References

References


• Ziqi Wang, Gu Xu, Hang Li and Ming Zhang, A Fast and Accurate Method for Approximate String Search, In ACL-HLT’11, pages 52-61, 2011.


Coffee Break
Outline of Tutorial

• Semantic Matching between Query and Document

• Approaches to Semantic Matching
  1. Matching by Query Reformulation
  2. Matching with Term Dependency Model
  3. Matching with Translation Model
  4. Matching with Topic Model
  5. Matching with Latent Space Model

• Summary
Matching based on Term Dependency

• Matching of consecutive terms in query and document indicates higher relevance
  – “hot dog”
  – “hot dog” ≠ hot + dog

• Query: order is quite free, but not completely free
  – “hot dog recipe”, “recipe hot dog”
  – “hot recipe dog” ×

• Term dependency: a sequence of terms representing soft query segmentation
Factors of Term Dependency

- **# terms**: number of terms in n-gram
  - 1 term (unigram)
  - Multiple terms (bigram, bi-terms ...)
- **Order**: order of terms is free or not
  - N-gram
  - Unordered N-terms
- **Skip**: maximum number of terms skipped within n-gram
  - No skip
  - $S$ skips
- Different choices of factors lead to different types of term dependencies
Types of Term Dependency

• Term dependency in query
  – Noun phrases (Bendersky & Croft, ’08)
  – Phrases & proximities (Bendersky & Croft, ’10; Shi & Nie, ’10; Bendersky & Croft, ‘12)

• Latent term dependency
  – Pseudo relevance feedback (Cao et al., ’08; Metzler & Croft ’07; Lease ’08; Bendersky et al., ’11)
  – Query expansion (Metzler ’11)
Addressing Term Mismatch based on Term Dependency

• Explicit term dependency represents degree of matching between query and document
  – Document including “hot dog” has higher matching degree than document including “hot” and “dog”

• Latent term dependency uses relations with additional terms to help ‘infer’ degree of matching
  – Query “airplane” has nonzero matching score with document including “aircraft”
Methods of Matching with Term Dependencies

• Term dependencies using Markov Random Fields (MRF)
  – Explicit term dependencies (Metzler & Croft, ’05)
  – Latent term dependencies (Metzler & Croft, 2008; Bendersky et al, ’11)
  – Weighted term dependencies (Bendersky et al., ’10; Bendersky et al, ’11)

• Extended IR models (Bendersky & Croft, ’12; Shi & Nie, ’10)
Markov Random Fields (MRF)

- Joint probability distribution represented by an undirected graph
  - Nodes: random variables
  - Edges: probabilistic dependencies
  - Cliques: subset of nodes such that every two nodes are connected

- Factorization of joint probability based on cliques

\[
P(x_1, \ldots, x_N) = \frac{1}{Z} \prod_{c \in \text{clique}(G)} \psi(c)
\]

- Normalizing factor
- Potential function
Modeling Term Dependencies with MRF (Metzler & Croft, 2005)

- **Nodes**
  - Document node
  - One node for each query term

- **Edges**
  - Each query node is linked with document node
  - Dependent terms are linked together
Modeling Term Dependencies with MRF

• Cliques
  – Representing how query terms are matched in document
  – Matching scores determined by potential function

• Joint probability
\[
P(q, d) = \frac{1}{Z} \prod_{c \in \text{clique}(G)} \exp(\lambda_c f(c))
\]

• Matching function
\[
F(q, d) = \sum_{c \in \text{clique}(G)} \lambda_c f(c)
\]
Modeling Term Dependencies with MRF

• Three types of feature functions \( f(c) \)
  
  – Fully independent
  \[
  f_1(q_i, d) = \log \left[ (1 - \alpha) \frac{tf(q_i, d)}{|d|} + \alpha \frac{cf(q_i)}{|C'|} \right]
  \]

  – Sequentially dependent
  \[
  f_2(q_i, \ldots, q_{i+k}, d) = \log \left[ (1 - \alpha) \frac{tf(q_i, \ldots, q_{i+k}, d)}{|d|} + \alpha \frac{cf(q_i, \ldots, q_{i+k})}{|C'|} \right]
  \]

  – Fully dependent
  \[
  f_3(q_i, \ldots, q_j, d) = \log \left[ (1 - \alpha) \frac{tf(q_i, \ldots, q_j, d)}{|d|} + \alpha \frac{cf(q_i, \ldots, q_j)}{|C'|} \right]
  \]
### Experimental Results

<table>
<thead>
<tr>
<th></th>
<th>fully independent</th>
<th>sequentially dependent</th>
<th>fully dependent</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAP</td>
<td>P@10</td>
<td>MAP</td>
</tr>
<tr>
<td>AP</td>
<td>0.1775</td>
<td>0.2912</td>
<td>0.1867*</td>
</tr>
<tr>
<td>WSJ</td>
<td>0.2592</td>
<td>0.4327</td>
<td>0.2776*</td>
</tr>
<tr>
<td>WT10g</td>
<td>0.2032</td>
<td>0.2866</td>
<td>0.2167*</td>
</tr>
<tr>
<td>GOV2</td>
<td>0.2502</td>
<td>0.4837</td>
<td>0.2832*</td>
</tr>
</tbody>
</table>

- Sequentially dependent and fully dependent outperform the baseline of fully independent
MRF Extensions

• Latent Term Dependencies (Metzler & Croft, 2007)
  – Latent terms exist behind query
  – E.g., collecting terms by pseudo relevance feedback

\[
\begin{align*}
d \quad & \quad q_1 \quad q_2 \quad q_3 \quad e_1 \quad e_2 \\
& \frac{\text{IDF}(q_1)}{\text{IDF}(q_2)} \quad \frac{\text{IDF}(q_3)}{\lambda(q_1q_2, d)} \quad \frac{\text{IDF}(q_3)}{\lambda(q_2q_3, d)}
\end{align*}
\]

• Weighted Term Dependencies (Bendersky et al., 2010)
  – High weights for most discriminative term dependencies (like IDF for unigram)
  – Leveraging different data resources such as web N-gram, Wikipedia etc. for estimating weights
Extended IR Model

• IR model as asymmetric kernels (Xu et al., ‘10)

\[
\text{BM25-Kernel}(q, d) = \sum_t \text{BM25-Kernel}_t(q, d)
\]

\[
\text{BM25-Kernel}_t(q, d) = \sum_x \text{IDF}_t(x) \times \frac{(k_3 + 1) \times f_t(x, q)}{k_3 + f_t(x, q)}
\times \frac{(k_1 + 1) \times f_t(x, d)}{k_1 \left(1 - b + b \frac{f_t(x, d)}{\text{avg} f_t} \right) + f_t(x, d)}
\]

• Dependency language model (Gao et al., ‘04)
  – Generate linkage \( l \) according to \( P(l|d) \)
  – Generate \( q \) according to \( P(q|l, d) \)

\[
P(q|d) = \sum_l P(q, l|d) = \sum_l P(l|d)P(q|l, d)
\]
References

References

Outline of Tutorial

• Semantic Matching between Query and Document
• Approaches to Semantic Matching
  1. Matching by Query Reformulation
  2. Matching with Term Dependency Model
  3. Matching with Translation Model
  4. Matching with Topic Model
  5. Matching with Latent Space Model
• Summary
Outline

• Statistical Machine Translation
• Search as Translation
• Methods for Matching with Translation Models
Statistical Machine Translation (SMT)

• Given sentence $C = c_1c_2 \cdots c_J$ in source language, translates it into sentence $E = e_1e_2 \cdots e_I$ in target language

$$E^* = \arg \max_E P(E|C)$$

$$= \arg \max_E \frac{P(C|E)P(E)}{P(C)}$$

$$= \arg \max_E P(C|E)P(E)$$
Typical Translation Models

• Word-based
  – Translating word to word

• Phrase-based
  – Translating based on phrase

• Syntax-based
  – Translating based on syntactic structure
Word-based Model: IBM Model One
(Brown et al., 1993)

• Generating target sentence
  – Choose the length of target language $I$, according to $P(I|C)$
  – For each position, $i (i = 1, 2, \ldots, I)$
    • Choose position $j$ in source sentence $C$ according to $P(j|C)$
    • Generate target word $e_i$ according to $P(e_i|c_i)$

$$P(E|C) = \frac{\epsilon}{(J+1)^I} \prod_{i=1}^{I} \sum_{j=1}^{J} P(e_i|c_j)$$
Model of Query Generation and Retrieval

- Task of retrieval: find the a posteriori most likely documents given query

\[
P(d|q, \mathcal{U}) = \frac{P(q|d, \mathcal{U}) \cdot P(d|\mathcal{U})}{P(q|\mathcal{U})}
\]

- query dependent
- query independent
Matching with Translation Model

- Translating document $d$ to query $q$
- Given query $q$ and document $d$, translation probability is viewed as matching score between $q$ and $d$
Addressing Term Mismatch with Translation Model

- Translation probability $P(q|w)$ represents matching degree between words in query and document.

| $q$   | $P(q|w)$  | $q$     | $P(q|w)$  |
|-------|-----------|---------|-----------|
| titanic | 0.56218   | Vista   | 0.80575   |
| ship   | 0.01383   | Windows | 0.05344   |
| movie  | 0.01222   | Download| 0.00728   |
| pictures | 0.01211 | ultimate| 0.00571   |
| sink   | 0.00697   | xp      | 0.00355   |
| facts  | 0.00689   | microsoft| 0.00342  |
| photos | 0.00533   | bit     | 0.00286   |
| rose   | 0.00447   | compatible| 0.00270  |
| people | 0.00441   | premium | 0.00244   |
| survivors | 0.00369 | free    | 0.00211   |

$w =$ titanic $\quad w =$ vista
Issues Need to be Addressed

- Self-translation probability $P(w|w)$
  - Both source language and target language are in the same language
  - Too large: decrease effect of using translation
  - Too small: direct matching less effective and hurt the performance of matching
Issues Need to be Addressed

• Training data
  – Synthetic data (Berger & Lafferty, ’99)
  – Document collection (Karimzadehgan & Zhai, ’10)
  – Title-body pairs of documents (Jin et al., ’02)
  – Query-title pairs in click-through data (Gao et al., ’10)
Issues Need to be Addressed

• Document fields
  – Use of title is better than body (Huang et al., ’10)
  – Titles and queries have similar languages
  – Bodies and queries have very different languages

\[
\text{Perplexity}(\tilde{P}, Q) = 2^{H(\tilde{P}, Q)} = 2^{-\sum_s \tilde{p}_s \log q_s}
\]
Methods for Matching with Translation

• Translating document to query
  – Word-based model (Berger & Lafferty, ’99; Gao et al., ’10)
  – Phrase-based model (Gao et al., ’10)
  – Syntax-based model (Park et al., ’11)
  – Topic-based model (Gao et al., ’11)
  – Learning translation probabilities from documents (Karimzadehgan & Zhai, ’10)

• Translating document model to query model
  – Translated query language model model (Jin et al., ’02)
Methods of Matching with Translation

• Basic model (Berger & Lafferty, ’99)

\[
P(q|d) = \frac{P(m|d)}{(n+1)^m} \prod_{j=1}^{m} \sum_{i=0}^{n} P(q_j|d_i)
\]

\[
= P(m|d) \prod_{j=1}^{m} \left( \frac{n}{n+1} P(q_j|d) + \frac{1}{n+1} P(q_j|\langle \text{null} \rangle) \right)
\]

Word \(q_j\) being translated from document \(d\).

\(P(q_j|d) = \sum_{w \in d} P(q_j|w)Q(w|d)\)

\(P(q_j|w)\): probability of \(w\) being translated to \(q_j\)

\(Q(w|d)\): un-smoothed document language model

Smoothing to avoid zero probability

• Adding self-translation (Gao et al., ‘10)

\[
P'(q_j|d) = \beta Q(q_j|d) + (1 - \beta) \sum_{w \in d} P(q_j|w)Q(w|d)
\]

Un-smoothed document language model
Performances of Word-based Translation Model in Search

<table>
<thead>
<tr>
<th></th>
<th>NDCG@1</th>
<th>NDCG@3</th>
<th>NDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM25 (baseline)</td>
<td>0.3181</td>
<td>0.3413</td>
<td>0.4045</td>
</tr>
<tr>
<td>WTM (without self-translation)</td>
<td>0.3210</td>
<td>0.3512</td>
<td>0.4211</td>
</tr>
<tr>
<td>WTM (with self-translation)</td>
<td>0.3310</td>
<td>0.3566</td>
<td>0.4232</td>
</tr>
</tbody>
</table>

- Evaluation based on 12071 real queries
- WTM can outperform baseline of BM25
- WTM can be further improved by self-translation
### Examples of Translation Probabilities

| $q$       | $P(q|w)$   | $q$       | $P(q|w)$   |
|-----------|------------|-----------|------------|
| titanic   | 0.56218    | Vista     | 0.80575    |
| ship      | 0.01383    | Windows   | 0.05344    |
| movie     | 0.01222    | Download  | 0.00728    |
| pictures  | 0.01211    | ultimate  | 0.00571    |
| sink      | 0.00697    | xp        | 0.00355    |
| facts     | 0.00689    | microsoft | 0.00342    |
| photos    | 0.00533    | bit       | 0.00286    |
| rose      | 0.00447    | compatible| 0.00270    |
| people    | 0.00441    | premium   | 0.00244    |
| survivors | 0.00369    | free      | 0.00211    |

$w = \text{titanic}$

$w = \text{vista}$
References

References

Outline of Tutorial

• Semantic Matching between Query and Document
• Approaches to Semantic Matching
  1. Matching by Query Reformulation
  2. Matching with Term Dependency Model
  3. Matching with Translation Model
  4. Matching with Topic Model
  5. Matching with Latent Space Model
• Summary
Outline

• Topic Models
• Methods of Matching with Topic Model
Topic Modeling

• Input
  – Document collection

• Processing
  – Discover latent topics in document collection

• Output
  – Latent topics in document collection
  – Topic representations of documents
Two Approaches

• Probabilistic approach

• Non-probabilistic approach

\[ \text{word} \rightarrow \text{D} \quad \approx \quad \text{word} \leftarrow \text{U} \times \text{VT} \]
Topic Modeling: Two Approaches (cont’)

• Probabilistic Topic Models
  – Model: probabilistic model (graphical model)
  – Learning: maximum likelihood estimation
  – Methods: PLSI, LDA

• Non-probabilistic Topic Models
  – Model: vector space model
  – Learning: matrix factorization
  – Methods: LSI, NMF, RLSI

• Non-probabilistic models can be reformulated as probabilistic models
Probabilistic Topic Model

• Topic: probability distribution over words
• Document: probability distribution over topics
• Graphical model
  – Word, topic, document, and topic distribution are represented as nodes
  – Probabilistic dependencies are represented as directed edges
  – Generation process
• Interpretation: soft clustering
Probabilistic Latent Semantic Indexing (Hofmann 1999)

1. select a document \(d\) from the collection with probability \(P(d)\)
2. for each document \(d\) in the collection
   
   (a) select a latent topic \(z\) with probability \(P(z|d)\)
   (b) generate a word \(w\) with probability \(P(w|z)\)
Latent Dirichlet Allocation
(Blei et al., 2003)

1. for each topic $k = 1, \cdots, K$
   
   (a) draw word distribution $\phi_k$ according to $\phi_k|\beta \sim Dir(\beta)$

2. for each document $d$ in the collection

   (a) draw topic distribution $\theta$ according to $\theta|\alpha \sim Dir(\alpha)$

   (b) for each word $w$ in the document $d$

      i. draw a topic $z$ according to $z|\theta \sim Mult(\theta)$

      ii. draw a word $w$ according to $w|z, \phi_{1:K} \sim Mult(\phi_z)$
Non-probabilistic Topic Model

• Document: vector of words
• Topic: vector of words
• Document representation: combination of topic vectors
• Matrix factorization
• Interpretation: projection to topic space
Latent Semantic Indexing (Deerwester et al., 1990)

- Representing document collection with co-occurrence matrix (TF or TFIDF)
- Performing Singular Value Decomposition (SVD) and producing k-dimensional topic space
Nonnegative Matrix Factorization
(Le and Seung, 2001)

• $U$ and $V$ are nonnegative

\[
\min_{U,V} \| D - UV^T \|_F \\
\text{s.t. } u_{ij} \geq 0; v_{ij} \geq 0
\]
Regularized Latent Semantic Indexing (Wang et al., 2011)

• Topics are sparse

\[
\min_{\mathbf{U}, \mathbf{V}} \sum_{n=1}^{N} \| \mathbf{d}_n - \mathbf{U} \mathbf{v}_n \|_2^2 + \lambda_1 \sum_{k=1}^{K} \| \mathbf{u}_k \|_1 + \lambda_2 \sum_{n=1}^{N} \| \mathbf{v}_n \|_2^2
\]

topics are sparse
Probabilistic Interpretation of Nonprobabilistic Models (RLSI)

\[
\begin{align*}
\min_{U,V} & \sum_{n=1}^{N} \|d_n - U v_n\|_2^2 + \lambda_1 \sum_{k=1}^{K} \|u_k\|_1 + \lambda_2 \sum_{n=1}^{N} \|v_n\|_2^2 \\
\end{align*}
\]

- Document generated according to Gaussian distribution
  \[P(d_n | U, v_n) \propto \exp(-\|d_n - U v_n\|_2^2)\]
- Laplacian prior
  \[P(u_k) \propto \exp(-\lambda_1 \|u_k\|_1)\]
- Gaussian prior
  \[P(v_n) \propto \exp(-\lambda_2 \|v_n\|_2^2)\]
Deal with Term Mismatch with Topic Model

- Topics of query and document are identified
- Match query and document through topics, although query and document do not share terms
- Linear combination with term model

\[ s(q, d) = \alpha s_{\text{topic}}(q, d) + (1 - \alpha) s_{\text{term}}(q, d) \]
Methods of Matching Using Topic Model

• Topic matching
  – Probabilistic model: PLSI (Hofmann ’99), LDA (Blei et al., ’03)
  – Non-probabilistic model: LSI (Deerwester et al., ’88), NMF (Lee & Seung ’00), RLSI (Wang et al., ’11), GMF (Wang et al., ’12)

• Smoothing
  – Clustering-based (Kurland & Lee ’04, Diaz ’05)
  – LDA-based (Wei & Croft ’06)
  – PLSI-based (Yi & Allan ’09)
Topic Level Matching

- Representing query and document as topic vectors (or topic distributions)
- Calculating matching score in topic space
Topic Level Matching (cont’)

• In RLSI, query and document representation
  
  \[- q \rightarrow v_q = (U^TU + \lambda_2 I)^{-1} q \]
  
  \[- d \rightarrow v_d = (U^TU + \lambda_2 I)^{-1} d \]

• Topic level matching
  
  – Cosine similarity
    
    \[ s_{\text{topic}}(q, d) = \frac{\langle v_q, v_d \rangle}{\|v_q\|_2 \|v_d\|_2} \]
  
  – Symmetric KL-divergence
    
    \[ s_{\text{topic}}(q, d) = 1 - \frac{1}{2} (KL(v_q\|v_d) + KL(v_d\|v_q)) \]
Experimental Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MAP</th>
<th>NDCG@1</th>
<th>NDCG@3</th>
<th>NDCG@5</th>
<th>NDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM25</td>
<td>0.3918</td>
<td>0.4400</td>
<td>0.4268</td>
<td>0.4298</td>
<td>0.4257</td>
</tr>
<tr>
<td>BM25+LSI</td>
<td>0.3952</td>
<td>0.4720</td>
<td>0.4410</td>
<td>0.4360</td>
<td>0.4365</td>
</tr>
<tr>
<td>BM25+NMF</td>
<td>0.3985*</td>
<td>0.4600</td>
<td>0.4445*</td>
<td>0.4408*</td>
<td>0.4347*</td>
</tr>
<tr>
<td>BM25+PLSI</td>
<td>0.3928</td>
<td>0.4680</td>
<td>0.4383</td>
<td>0.4351</td>
<td>0.4291</td>
</tr>
<tr>
<td>BM25+LDA</td>
<td>0.3952</td>
<td>0.4760*</td>
<td>0.4478*</td>
<td>0.4332</td>
<td>0.4292</td>
</tr>
<tr>
<td>BM25+RLSI</td>
<td>0.3998*</td>
<td>0.4800*</td>
<td>0.4461*</td>
<td>0.4498*</td>
<td>0.4420*</td>
</tr>
</tbody>
</table>

- Topic models can improve the baseline of BM25
- LDA, NMF, and RLSI perform slightly better than the others
References

References

Outline of Tutorial

• Semantic Matching between Query and Document
• Approaches to Semantic Matching
  1. Matching by Query Reformulation
  2. Matching with Term Dependency Model
  3. Matching with Translation Model
  4. Matching with Topic Model
  5. Matching with Latent Space Model
• Summary
Matching in Latent Space

• Motivation
  – Matching between query and document in latent space

• Assumption
  – Queries have similarity
  – Documents have similarity
  – Click-through data represent “similarity” relations between queries and documents

• Approach
  – Projection to latent space
  – Regularization or constraints

• Results
  – Significantly enhance accuracy of query document matching
Matching in Latent Space

Latent Space

\[ \begin{align*}
L_q & : q_1 \rightarrow \mathbb{R}^q \\
L_d & : d_1 \rightarrow \mathbb{R}^d
\end{align*} \]
IR Models as Similarity Functions (Xu et al., 2010)

Mapping functions are diagonal matrices

VSM, BM25, LM, MRF

Query Space

Document Space
IR Models as Similarity Functions

- **VSM**

\[
f_{\text{VSM}}(q, d) = \langle \phi_{\text{VSM}}(q), \phi'_{\text{VSM}}(d) \rangle = \langle q, d \rangle.
\]

- **BM25**

\[
f_{\text{BM25}}(q, d) = \langle \phi_{\text{BM25}}(q), \phi'_{\text{BM25}}(d) \rangle
\]

\[
\phi_{\text{BM25}}(q)_x = \frac{(k_3 + 1) \cdot f(x, q)}{k_3 + f(x, q)}
\]

\[
\phi'_{\text{BM25}}(d)_x = \text{IDF}(x) \cdot \frac{(k_1 + 1) \cdot f(x, d)}{k_1 \left(1 - b + b \frac{f(d)}{\text{avg}f}\right) + f(x, d)}
\]
Deal with Term Mismatch with Latent Space Model

• Matching in Latent Space can solve the problem by
  – Reducing dimensionality of latent space (from term level matching to semantic matching)
  – Correlating semantically similar terms (matrices are not diagonal)
  – Automatically learning mapping functions from data

• Generalized and Learnable of IR models
Partial Least Square (PLS)

• Input
  – Training data: \{((q_i, d_i, c_i))_{1 \leq i \leq N}, q_i \in Q, d_i \in D, c_i \in \{+1, -1\} or c_i \in R \}

• Output
  – Similarity function \( f(q, d) \)

• Assumption
  – Two linear and orthonormal transformations \( L_q \) and \( L_d \)
  – Dot product as similarity function \( f(q, d) = \langle L_q \cdot q, L_d \cdot d \rangle \)

• Optimization
  \[
  \arg \max_{L_q, L_d} = \sum_{(q_i, d_i)} c_i f(q_i, d_i),
  \]
  \[
  L_q L_q^T = I, \quad L_d L_d^T = I
  \]
Solution of Partial Least Square

• Non-convex optimization
• Can prove that global optimal solution exists
• Global optimal can be found by solving SVD
• SVD of matrix $M_S - M_D = U\Sigma V^T$
Regularized Mapping to Latent Space (Wu et al., ‘13)

• Input
  – Training data: \( \{(q_i, d_i, c_i)\}_{1 \leq i \leq N}, q_i \in Q, d_i \in D, c_i \in \{+1, -1\} \) or \( c_i \in R \)

• Output
  – Similarity function \( f(q, d) \)

• Assumption
  – \( \ell_1 \) and \( \ell_2 \) regularization on \( L_X \) and \( L_Y \) (sparse transformations)
  – Dot product as similarity function \( f(q, d) = \langle L_q \cdot q, L_d \cdot d \rangle \)

• Optimization

\[
\arg \max_{L_q, L_d} = \sum_{(q_i, d_i)} c_i f(q_i, d_i),
\]

\[
|l_q| \leq \theta_q, \quad |l_d| \leq \theta_d, \quad \|l_q\| \leq \tau_q, \quad \|l_d\| \leq \tau_d
\]
Solution of Regularized Mapping to Latent Space

- Coordinate Descent
- Repeat
  - Fix $L_X$, update $L_Y$
  - Fix $L_Y$, update $L_X$
- Update can be parallelized by rows
Bilingual Topic Model
(Gao et al., ‘11)

- A natural extension of LDA for generating pairs of documents
- Each query document pair is generated from the same distribution of topics
- EM algorithm can be employed to estimate the parameters

\[
P(q|d) = \prod_{q \in q} P_{bltm}(q|d) = \prod_{q \in q} \sum_{z} P(q|\phi_{q}^{z}) P(z|\theta^{d})
\]
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>PLS</th>
<th>RMLS</th>
<th>BLTM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Assumption</strong></td>
<td>Orthogonal</td>
<td>$\ell_1$ and $\ell_2$</td>
<td>Topic Modeling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>regularization</td>
<td></td>
</tr>
<tr>
<td><strong>Optimization Method</strong></td>
<td>Singular Value</td>
<td>Coordinate</td>
<td>EM</td>
</tr>
<tr>
<td></td>
<td>Decomposition</td>
<td>Descent</td>
<td></td>
</tr>
<tr>
<td><strong>Optimality</strong></td>
<td>Global optimum</td>
<td>Local optimum</td>
<td>Local optimum</td>
</tr>
<tr>
<td><strong>Efficiency</strong></td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>
Experimental Results

Table 7.1: Performances of latent space models in search.

<table>
<thead>
<tr>
<th>Model</th>
<th>NDCG@1</th>
<th>NDCG@3</th>
<th>NDCG@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM25 (baseline)</td>
<td>0.637</td>
<td>0.690</td>
<td>0.690</td>
</tr>
<tr>
<td>SSI</td>
<td>0.538</td>
<td>0.621</td>
<td>0.629</td>
</tr>
<tr>
<td>SVDFeature</td>
<td>0.663</td>
<td>0.720</td>
<td>0.727</td>
</tr>
<tr>
<td>BLTM</td>
<td>0.657</td>
<td>0.702</td>
<td>0.701</td>
</tr>
<tr>
<td>PLS</td>
<td>0.676</td>
<td>0.728</td>
<td>0.736</td>
</tr>
<tr>
<td>RMLS</td>
<td>0.686</td>
<td>0.732</td>
<td>0.729</td>
</tr>
</tbody>
</table>

- 94,022 queries, 111,631 documents, and click through data;
- RMLS and PLS work better than BM25, SSI, SVDFeature, and BLTM
- RMLS works equally well as PLS, with higher learning efficiency and scalability
Learning Semantic Embedding using the DSSM

**Initialization:**

Neural networks are initialized with random weights

---

Semantic vector $v_s$

- $d=300$
- $W_4$
- $d=500$
- $W_3$
- $d=500$
- $W_2$
- $\text{dim} = 50K$
- $W_1$
- $\text{dim} = 5M$

Input word/phrase $s$: “racing car”

Letter-trigram embedding matrix

Letter-trigram enco. matrix (fixed)

Bag-of-words vector $v_{t^+}$

- $d=300$
- $d=500$
- $\text{dim} = 50K$

Input word/phrase $t^+$: “formula one”

Bag-of-words vector $v_{t^-}$

- $d=300$
- $d=500$
- $\text{dim} = 50K$

Input word/phrase $t^-$: “ford model t”

---

From Jianfeng Gao, CIKM 2014
Learning Semantic Embedding using the DSSM

**Training (Back Propagation):**

Compute Cosine similarity between semantic vectors

\[
\frac{\partial \sum_{t'=[t^+, t^-]} \exp(\cos(v_s, v_{t'}))}{\partial W}
\]

Semantic vector

- **Semantic vector**
  - \( v_s \) with \( d=300 \)
  - \( v_{t^+} \) with \( d=300 \)
  - \( v_{t^-} \) with \( d=300 \)

**Letter-trigram embedding matrix**

- \( W_4 \) with \( d=500 \)
- \( W_3 \) with \( d=500 \)
- \( W_2 \) with \( d=500 \)
- \( W_1 \) with \( \text{dim} = 50K \)

**Letter-trigram enco. matrix (fixed)**

- \( \text{dim} = 50K \)

**Bag-of-words vector**

- \( \text{dim} = 5M \)

**Input word/phrase**

- \( s: \text{“racing car”} \)
- \( t^+: \text{“formula one”} \)
- \( t^-: \text{“ford model t”} \)

[From Jianfeng Gao, CIKM 2014]
Learning Semantic Embedding using the DSSM

After training converged:

Cosine similarity between semantic vectors

Semantic vector

d=300

d=500

W4

W3

W2

dim = 50K

dim = 50K

Letter-trigram embedding matrix

Letter-trigram enco. matrix (fixed)

Bag-of-words vector

Input word/phrase

"racing car"

"formula one"

"ford model t"

[Huang, He, Gao, Deng, Acero, Heck, 2013]
Experimental Results

Table 7.2: Performances of latent space models in search.

<table>
<thead>
<tr>
<th>Model</th>
<th>NDCG@1</th>
<th>NDCG@3</th>
<th>NDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM25 (baseline)</td>
<td>0.308</td>
<td>0.373</td>
<td>0.455</td>
</tr>
<tr>
<td>WTM</td>
<td>0.332</td>
<td>0.400</td>
<td>0.478</td>
</tr>
<tr>
<td>LSI</td>
<td>0.298</td>
<td>0.372</td>
<td>0.455</td>
</tr>
<tr>
<td>PLSI</td>
<td>0.295</td>
<td>0.371</td>
<td>0.456</td>
</tr>
<tr>
<td>BLTM</td>
<td>0.337</td>
<td>0.403</td>
<td>0.480</td>
</tr>
<tr>
<td>DSSM (linear)</td>
<td>0.357</td>
<td>0.422</td>
<td>0.495</td>
</tr>
<tr>
<td>DSSM (non-linear)</td>
<td><strong>0.362</strong></td>
<td><strong>0.425</strong></td>
<td><strong>0.498</strong></td>
</tr>
</tbody>
</table>

- Experiments conducted with 16510 queries, and each query on average associated with 15 webpages
- DSSM outperformed all baselines
- DSSM (non-linear) is the best
References


• Roman Rosipal and Nicole Krämer. Overview and recent advances in partial least squares. In SLSFS’05, pages 34–51, 2006.


Outline of Tutorial

• Semantic Matching between Query and Document

• Approaches to Semantic Matching
  1. Matching by Query Reformulation
  2. Matching with Term Dependency Model
  3. Matching with Translation Model
  4. Matching with Topic Model
  5. Matching with Latent Space Model

• Summary
Summary of Tutorial

• Query document matching is one of the biggest challenge in search
• Machine learning for matching between query and document is making progress
• Matching at form, phrase, sense, topic, and structure aspects
• General problem: learning to match
Approaches

- Matching by query reformulation
- Matching with term dependency model
- Matching with translation model
- Matching with topic model
- Matching with latent space model
## Characteristics of Approaches

<table>
<thead>
<tr>
<th>Model</th>
<th>Training Data</th>
<th>Complexity of Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>query</td>
<td>search log</td>
</tr>
<tr>
<td>Dependency</td>
<td>query-document</td>
<td>relevance</td>
</tr>
<tr>
<td>Translation</td>
<td>query-document</td>
<td>click-through</td>
</tr>
<tr>
<td>Topic</td>
<td>document</td>
<td>document</td>
</tr>
<tr>
<td>Latent</td>
<td>query-document</td>
<td>click-through</td>
</tr>
</tbody>
</table>
Open Problems

• Topic drift: language is by nature synonymous and polysemous
• Scalability: e.g., topic model and latent space model needs large scale computing environment
• Missing information in training data: for rare queries and documents
• More NLP techniques is needed: for long queries and NLP queries
• Evaluation measures: Current approaches has limitation
Q & A
Thank you!

junxu@ict.ac.cn