Machine Learning and Applications to Social Media

Hang Li
Huawei Technologies
Outline

1. Introduction
2. Learning to Rank
3. Learning to Match
4. Applications to Social Media
5. Summary
Two Challenges: Information Overload and Information Shortage
Video: Weibo Robot
Main Features of Weibo Robot V1

• Features Developed
  – Following People
  – Re-Tweeting (Forwarding Tweets)
  – Generating Short Comments

• Features To Be Developed
  – Generating Original Tweets
  – Generating Long Comments
微软亚洲研究院终于有60000位粉丝啦。

你们的每一条转发和评论主页君都会认真浏览。
亲爱的@小诺_pinocchio @明天你是top @蓝色梦旅人
@谢涛TaoXie，
你们有着最高的互动率，想必就是传说中的“真爱粉”了！

正如知名IT评论人炳叔所说：不公知不五毛不卖萌不传销不
淘宝，搞到6万粉丝，还真是难啊。
2. Learning to Rank
2.1. Overview of Learning to Rank
Ranking Plays Key Role in Many Applications
Ranking Problem:
Example = Document Retrieval

$D = \{d_1, d_2, \ldots, d_N\}$

query $q$

ranking of documents

ranking based on relevance, importance, preference

$f(q,d)$

$d_{q,1}$

$d_{q,2}$

\vdots

$d_{q,n_q}$
# Ranking Problem

Example = Recommenders System

<table>
<thead>
<tr>
<th></th>
<th>Item1</th>
<th>Item2</th>
<th>Item3</th>
<th>...</th>
<th>ItemN</th>
</tr>
</thead>
<tbody>
<tr>
<td>User1</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User2</td>
<td>1</td>
<td></td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>?</td>
<td></td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>UserM</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Ranking Problem
Example = Machine Translation

sentence source language

\[ f \]

Generative Model

\[ e_1 \]
\[ e_2 \]
\[ e_{1000} \]

ranked sentence candidates in target language

Re-Ranking Model

\[ \tilde{e} \]

re-ranked top sentence in target language
Ranking Problem
Example = Meta Search
Learning to Rank

• Definition 1 (in broad sense)
  Learning to rank = any machine learning technology for ranking problem

• Definition 2 (in narrow sense)
  Learning to rank = machine learning technology for ranking creation and ranking aggregation

• This tutorial takes Definition 2
Taxonomy of Problems in Learning to Rank

Learning to Rank

- Ranking Creation
  - Supervised (e.g., Ranking SVM)
  - Unsupervised (e.g., BM25)

- Ranking Aggregation
  - Supervised (e.g., CRank)
  - Unsupervised (e.g., Borda Count)
Ranking Creation
(with Local Ranking Model)

requests

\[ S = \{q_1, q_2, \ldots, q_i, \ldots, q_M\} \]

objects

\[ O = \{o_1, o_2, \ldots, o_j, \ldots, o_N\} \]

objects

\[ O_i = \{o_{i,1}, o_{i,2}, \ldots, o_{i,n_i}\} \]

ranking of objects

\[ f(q, o) \]

\[ o_{i,1} \rightarrow f(q_i, o_{i,1}) \]

\[ o_{i,2} \rightarrow f(q_i, o_{i,2}) \]

\[ \vdots \]

\[ o_{i,n_i} \rightarrow f(q_i, o_{i,n_i}) \]
Learning for Ranking Creation

• Creating a ranking list of offerings based on request and offerings
• Feature-based
• Usually local ranking model
• Usually supervised learning
Ranking Aggregation
Learning for Ranking Aggregation

• Aggregating a ranking list from multiple ranking lists of offerings
• Ranking based
• Usually global ranking model
• Both supervised and unsupervised learning
2.2. Problem and Approaches of Learning to Rank
Ranking Problem:
Example = Document Search

documents

\[ D = \{d_1, d_2, \ldots, d_N\} \]

query
\[ q \]

\[ f(q, d) \]

ranking of documents
\[ d_{q,1} \]
\[ d_{q,2} \]
\[ \vdots \]
\[ d_{q,n_q} \]

ranking based on relevance, importance, preference
Traditional Approach = Probabilistic Model

\[
P(r | q, d) \quad R \in \{1, 0\}
\]

query \( q \)
documents \( d_1, d_2, \ldots, d_N \)
ranking of documents

\[
d_1 \sim P(r | q, d_1) \\
d_2 \sim P(r | q, d_2) \\
\vdots \\
d_n \sim P(r | q, d_n)
\]
BM25
[Robertson & Walker 94]

documents

$q$

query

$P(r \mid q, d)$

$R \in \{1,0\}$

ranking function

$$\sum_{w \in d \cap q} \frac{(k + 1)tf(w)}{(1 - b)k + b \frac{dl}{avgdl}} + tf(w)$$

ranking of documents

$$d_1 \sim P(r \mid q, d_1)$$

$$d_2 \sim P(r \mid q, d_2)$$

$$\vdots$$

$$d_n \sim P(r \mid q, d_n)$$
PageRank
[Page et al, 1999]

\[ P(d_i) = \alpha \sum_{d_j \in M(d_i)} \frac{P(d_j)}{L(d_j)} + (1 - \alpha) \frac{1}{n} \]
New Approach = Learning to Rank

\[
D = \{ d_1, d_2, \ldots, d_N \}
\]

\[
\begin{align*}
q_1 & \quad q_m \\
d_{1,1} & \quad d_{m,1} \\
d_{1,2} & \quad d_{m,2} \\
\vdots & \quad \vdots \\
d_{1,n_1} & \quad d_{m,n_m}
\end{align*}
\]

Learning System

\[
f(q, d)
\]

Ranking System

\[
f(q_{m+1}, d_{m+1,1}) \quad f(q_{m+1}, d_{m+1,2}) \quad \cdots \quad f(q_{m+1}, d_{m+1,n_{m+1}})
\]
Training Process

1. Data Labeling (rank)
\[
\begin{pmatrix}
    d_{1,1} \\
    d_{1,2} \\
    \vdots \\
    d_{1,n_1}
\end{pmatrix}
\]

2. Feature Extraction
\[
\begin{pmatrix}
    d_{1,1} & y_{1,1} \\
    d_{1,2} & y_{1,2} \\
    \vdots & \vdots \\
    d_{1,n_1} & y_{1,n_1}
\end{pmatrix}
\]

3. Learning
\[
\begin{pmatrix}
    x_{1,1} & y_{1,1} \\
    x_{1,2} & y_{1,2} \\
    \vdots & \vdots \\
    x_{1,n_1} & y_{1,n_1}
\end{pmatrix}
\]

\[
\begin{pmatrix}
    x_{m,1} & y_{m,1} \\
    x_{m,2} & y_{m,2} \\
    \vdots & \vdots \\
    x_{m,n_m} & y_{m,n_m}
\end{pmatrix}
\]

\[ f(x) \]
Testing Process

1. Data Labeling (rank)

\[
q_{m+1} = \begin{cases} 
  d_{m+1,1} \\
  d_{m+1,2} \\
  \vdots \\
  d_{m+1,n_{m+1}} 
\end{cases}
\]

2. Feature Extraction

\[
q_{m+1} = \begin{cases} 
  d_{m+1,1} \\
  d_{m+1,2} \\
  \vdots \\
  d_{m+1,n_{m+1}} 
\end{cases}
\]

\[
\begin{cases} 
  y_{m+1,1} \\
  y_{m+1,2} \\
  \vdots \\
  y_{m+1,n_{m+1}} 
\end{cases}
\]

3. Ranking with \( f(x) \)

\[
\begin{cases} 
  x_{m+1,1} \\
  x_{m+1,2} \\
  \vdots \\
  x_{m+1,n_{m+1}} 
\end{cases}
\]

\[
\begin{cases} 
  y_{m+1,1} \\
  y_{m+1,2} \\
  \vdots \\
  y_{m+1,n_{m+1}} 
\end{cases}
\]

4. Evaluation

\[
\begin{cases} 
  x_{m+1,1} \\
  x_{m+1,2} \\
  \vdots \\
  x_{m+1,n_{m+1}} 
\end{cases}
\]

\[
\begin{cases} 
  f(x_{m+1,1}) \\
  f(x_{m+1,2}) \\
  \vdots \\
  f(x_{m+1,n_{m+1}}) 
\end{cases}
\]

\[
\begin{cases} 
  y_{m+1,1} \\
  y_{m+1,2} \\
  \vdots \\
  y_{m+1,n_{m+1}} 
\end{cases}
\]

Evaluation Result
Notes

- Features are functions of query and document
- Query and associated documents form a group
- Groups are i.i.d. data
- Feature vectors within group are not i.i.d. data
- Ranking model is function of features
- Several data labeling methods (here labeling of grade)
Issues in Learning to Rank

• Data Labeling
• Feature Extraction
• Evaluation Measure
• Learning Method (Model, Loss Function, Algorithm)
Data Labeling Problem

• E.g., relevance of documents w.r.t. query
Data Labeling Methods

• Labeling of Grades
  – Multiple levels (e.g., relevant, partially relevant, irrelevant)
  – Widely used in IR

• Labeling of Ordered Pairs
  – Ordered pairs between documents (e.g. A>B, B>C)
  – Implicit relevance judgment: derived from click-through data

• Creation of List
  – List (or permutation) of documents is given
  – Ideal but difficult to implement
Implicit Relevance Judgment

ranking of documents at search system

Doc A

Doc B

users often clicked on Doc B

ordered pair

B > A

Doc C
Feature Extraction

Query

Doc A

Doc B

Doc C

Feature Vectors

BM25

PageRank

Query-document feature

Document feature
### Example Features

**Table 2.3: Example Features of Learning to Rank for Web Search**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Type</th>
<th>Explanation</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of occurrences</td>
<td>Matching</td>
<td>number of times query exactly occurs in title, anchor, URL, extracted title, associated query, and body</td>
<td>[90]</td>
</tr>
<tr>
<td>BM25</td>
<td>Matching</td>
<td>BM25 scores on title, anchor, URL, extracted title, associated query, and body</td>
<td>[109]</td>
</tr>
<tr>
<td>N-gram BM25</td>
<td>Matching</td>
<td>BM25 scores of n-grams on title, anchor, URL, extracted title, associated query, and body</td>
<td></td>
</tr>
<tr>
<td>Edit Distance</td>
<td>Matching</td>
<td>edit distance scores between query and title, anchor, URL, extracted title, associated query, and span in body (minimum length of text segment including all query words [94])</td>
<td>Our unpublished work</td>
</tr>
<tr>
<td>Number of in-links</td>
<td>Document</td>
<td>number of in-links to the page</td>
<td>[78]</td>
</tr>
<tr>
<td>PageRank</td>
<td>Document</td>
<td>importance score of page calculated on web link graph</td>
<td></td>
</tr>
<tr>
<td>Number of clicks</td>
<td>Document</td>
<td>number of clicks on the page in search log</td>
<td></td>
</tr>
<tr>
<td>BrowseRank</td>
<td>Document</td>
<td>importance score of page calculated on user browsing graph</td>
<td>[72]</td>
</tr>
<tr>
<td>Spam score</td>
<td>Document</td>
<td>likelihood of spam page</td>
<td>[45]</td>
</tr>
<tr>
<td>Page quality score</td>
<td>Document</td>
<td>likelihood of low quality page</td>
<td>[10]</td>
</tr>
</tbody>
</table>
Evaluation Measures

• Important to rank top results correctly

• Measures
  – NDCG (Normalized Discounted Cumulative Gain)
  – MAP (Mean Average Precision)
  – MRR (Mean Reciprocal Rank)
  – WTA (Winners Take All)
  – Kendall’s Tau
NDCG

- Evaluating ranking using labeled grades
- NDCG at position $j$

$$\frac{1}{n_j} \sum_{i=1}^{j} \frac{(2^{r(i)} - 1)}{\log(1 + i)}$$
NDCG (cont’)

• Example: perfect ranking
  – (3, 3, 2, 2, 1, 1, 1) grade $r=3,2,1$
  – (7, 7, 3, 3, 1, 1, 1) gain $2^{r(j)} - 1$
  – (1, 0.63, 0.5, 0.43, 0.39, 0.36, 0.33) position discount
  – (7, 11.41, 12.91, ...) DCG $\frac{1}{\log(1+ j)}$
  $$\sum_{i=1}^{j} (2^{r(i)} - 1) / \log(1+i)$$
  – (1/7, 1/11.41, 1/12.91, ...) normalizing factor $n_j$
  – (1, 1,1,1,1,1,1) NDCG for perfect ranking
NDCG (cont’)

• Example: imperfect ranking
  – (2, 3, 2, 3, 1, 1, 1)
  – (3, 7, 3, 7, 1, 1, 1) Gain
  – (1, 0.63, 0.5, 0.43, 0.39, 0.36, 0.33) Position discount
  – (3, 7.41, 8.91, ... ) DCG
  – (1/7, 1/11.41, 1/12.91, ...) normalizing factor
  – (0.43, 0.65, 0.69, .... ) NDCG

• Imperfect ranking decreases NDCG
Relations with Other Learning Tasks

- No need to predict category vs Classification
- No need to predict value of $f(q,d)$ vs Regression
- Relative ranking order is more important vs Ordinal regression
- Learning to rank can be approximated by classification, regression, ordinal regression
Ordinal Regression (Ordinal Classification)

• Categories are ordered
  – 5, 4, 3, 2, 1
  – e.g., rating restaurants

• Prediction
  – Map to ordered categories
Three Major Approaches

• Pointwise approach
• Pairwise approach
• Listwise approach

• SVM based
• Boosting based
• Neural Network based
• Others
### Table 2.6: Categorization of Learning to Rank Methods

<table>
<thead>
<tr>
<th>Category</th>
<th>SVM</th>
<th>Boosting</th>
<th>Neural Net</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Subset Ranking [29]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LambdaMART [102]</td>
<td>LambdaRank</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>[12]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ListNet [14]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SoftRank [95]</td>
</tr>
<tr>
<td></td>
<td>PermuRank [110]</td>
<td></td>
<td></td>
<td>AppRank [81]</td>
</tr>
</tbody>
</table>

Pointwise Approach

• Transforming ranking to regression, classification, or ordinal classification
• Query-document group structure is ignored
# Pointwise Approach

## Table 2.7: Characteristics of Pointwise Approach

### Pointwise Approach (Classification)

<table>
<thead>
<tr>
<th>Learning</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td></td>
</tr>
<tr>
<td>feature vector</td>
<td>feature vectors</td>
</tr>
<tr>
<td>( x )</td>
<td>( x = {x_i}_{i=1}^n )</td>
</tr>
<tr>
<td>Output</td>
<td></td>
</tr>
<tr>
<td>category</td>
<td>ranking list</td>
</tr>
<tr>
<td>( y = \text{classifier}(f(x)) )</td>
<td>( \text{sort}([f(x_i)]_{i=1}^n) )</td>
</tr>
<tr>
<td>Model Loss</td>
<td></td>
</tr>
<tr>
<td>( \text{classifier}(f(x)) )</td>
<td>( \text{ranking model} f(x) )</td>
</tr>
<tr>
<td></td>
<td>( \text{classification loss} )</td>
</tr>
</tbody>
</table>

### Pointwise Approach (Regression)

<table>
<thead>
<tr>
<th>Learning</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td></td>
</tr>
<tr>
<td>feature vector</td>
<td>feature vectors</td>
</tr>
<tr>
<td>( x )</td>
<td>( x = {x_i}_{i=1}^n )</td>
</tr>
<tr>
<td>Output</td>
<td></td>
</tr>
<tr>
<td>real number</td>
<td>ranking list</td>
</tr>
<tr>
<td>( y = f(x) )</td>
<td>( \text{sort}([f(x_i)]_{i=1}^n) )</td>
</tr>
<tr>
<td>Model Loss</td>
<td></td>
</tr>
<tr>
<td>regression model</td>
<td>ranking model ( f(x) )</td>
</tr>
<tr>
<td>( f(x) )</td>
<td>ranking loss</td>
</tr>
<tr>
<td>regression loss</td>
<td>ranking loss</td>
</tr>
</tbody>
</table>
Pointwise Approach

<table>
<thead>
<tr>
<th></th>
<th>Learning</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>feature vector</td>
<td>feature vectors</td>
</tr>
<tr>
<td></td>
<td>$x$</td>
<td>$x = {x_i}_{i=1}^n$</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>ordered category</td>
<td>ranking list</td>
</tr>
<tr>
<td></td>
<td>$y = \text{threshold}(f(x))$</td>
<td>$\text{sort}({f(x_i)}_{i=1}^n)$</td>
</tr>
<tr>
<td><strong>Model Loss</strong></td>
<td>threshold($f(x)$)</td>
<td>ranking model $f(x)$</td>
</tr>
<tr>
<td></td>
<td>ordinal classification loss</td>
<td>ranking loss</td>
</tr>
</tbody>
</table>

Pointwise Approach (Ordinal Classification)
Pairwise Approach

• Transforming ranking to pairwise classification
• Query-document group structure is ignored
### Table 2.8: Characteristics of Pairwise Approach

#### Pairwise Approach (Classification)

<table>
<thead>
<tr>
<th></th>
<th>Learning</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>feature vectors $x^{(1)}, x^{(2)}$</td>
<td>feature vectors $x = {x_i}_{i=1}^n$</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>pairwise classification $\text{classifier}(f(x^{(1)}) - f(x^{(2)}))$</td>
<td>ranking list $\text{sort}([f(x_i)]_{i=1}^n)$</td>
</tr>
<tr>
<td><strong>Model Loss</strong></td>
<td>$\text{classifier}(f(x))$</td>
<td>ranking model $f(x)$</td>
</tr>
<tr>
<td></td>
<td>pairwise classification loss</td>
<td>ranking loss</td>
</tr>
</tbody>
</table>

#### Pairwise Approach (Regression)

<table>
<thead>
<tr>
<th></th>
<th>Learning</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>feature vectors $x^{(1)}, x^{(2)}$</td>
<td>feature vectors $x = {x_i}_{i=1}^n$</td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td>pairwise regression $f(x^{(1)}) - f(x^{(2)})$</td>
<td>ranking list $\text{sort}([f(x_i)]_{i=1}^n)$</td>
</tr>
<tr>
<td><strong>Model Loss</strong></td>
<td>regression model $f(x)$</td>
<td>ranking model $f(x)$</td>
</tr>
<tr>
<td></td>
<td>pairwise regression loss</td>
<td>ranking loss</td>
</tr>
</tbody>
</table>
Listwise Approach

• List as instance
• Query-document group structure is used
• Straightforwardly represents learning to rank problem
Listwise Approach

<table>
<thead>
<tr>
<th>Input</th>
<th>Learning</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>(x = {x_i}_{i=1}^n)</td>
<td>(x = {x_i}_{i=1}^n)</td>
</tr>
<tr>
<td>Output</td>
<td>ranking list</td>
<td>ranking list</td>
</tr>
<tr>
<td></td>
<td>(\text{sort}([f(x_i)]_{i=1}^n))</td>
<td>(\text{sort}([f(x_i)]_{i=1}^n))</td>
</tr>
<tr>
<td>Model</td>
<td>ranking model (f(x))</td>
<td>ranking model (f(x))</td>
</tr>
<tr>
<td>Loss</td>
<td>listwise loss function</td>
<td>ranking loss</td>
</tr>
</tbody>
</table>

Table 2.9: Characteristics of Listwise Approach
Evaluation Results

• Pairwise approach and listwise approach perform better than pointwise approach
• LabmdaMART performs best in Yahoo Learning to rank Challenge
• No significant difference among pairwise and listwise methods
2.3. Methods of Learning to Rank
Ranking SVM
Pairwise Classification

• Converting document list to document pairs
Transforming Ranking to Pairwise Classification

- Input space: $X$
- Ranking function $f : X \rightarrow R$
- Ranking: $x_i \succ x_j \iff f(x_i; w) > f(x_j; w)$
- Linear ranking function: $f(x; w) = \langle w, x \rangle$
  $\langle w, x_i - x_j \rangle > 0 \iff f(x_i; w) > f(x_j; w)$
- Transforming to pairwise classification:
  $$(x_i - x_j, z), \quad y = \begin{cases} 
  +1 & x_i \succ x_j \\
  -1 & x_j \succ x_i 
  \end{cases}$$
Ranking Problem

\[ w \]

\[ w \]

\( x_1 \)

\( x_2 \)

\( x_3 \)

\( \triangle \) grade 3

\( \circ \) grade 2

\( \times \) grade 1
Transformed Pairwise Classification Problem

Positive Examples

Negative Examples

Redundant

\[ f(x; w) \]

\[ x_1 - x_3 \]

\[ x_2 - x_3 \]

\[ x_1 - x_2 \]

\[ x_3 - x_1 \]

\[ x_3 - x_2 \]
Ranking SVM
(Hebrich et al., 1999)

• Pairwise classification on differences of feature vectors
• Corresponding positive and negative examples
• Negative examples are redundant and can be discarded
• Hyper plane passes the origin
• Soft margin and kernel can be used
• *Ranking SVM* = pairwise classification SVM
Learning of Ranking SVM

\[
\begin{align*}
\min_{w, \xi} & \quad \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{N} \xi_i \\
y_i \langle w, x_i^{(1)} - x_i^{(2)} \rangle & \geq 1 - \xi_i \quad i = 1, \ldots, N \\
\xi_i & \geq 0
\end{align*}
\]

\[
\begin{align*}
\min_{w} \sum_{i=1}^{l} \left[ 1 - y_i \langle w, x_i^{(1)} - x_i^{(2)} \rangle \right] + \lambda \| w \|^2 \\
[s]_+ = \max(0, s) \quad \lambda = \frac{1}{2C}
\end{align*}
\]
IR SVM
Cost-sensitive Pairwise Classification

• Converting to document pairs

Query

grade 3
Doc A

grade 2
Doc B

grade 1
Doc C

Critical
Doc A

Doc B

Doc C

Not Critical
Doc B

Doc C

Doc C
Problems with Ranking SVM

• Not sufficient emphasis on correct ranking on top grades: 3, 2, 1
  ranking 1:  2 3 2 1 1 1 1
  ranking 2:  3 2 1 2 1 1 1
  ranking 2 should be better than ranking 1
  Ranking SVM views them as the same

• Numbers of pairs vary according to queries
  q1: 3 2 2 1 1 1 1
  q2: 3 3 2 2 2 1 1 1 1 1
  number of pairs for q1 : 2*(2-2) + 4*(3-1) + 8*(2-1) = 14
  number of pairs for q2: 6*(3-2) + 10*(3-1) + 15*(2-1) = 31
  Ranking SVM is biased toward q2
IR SVM
(Cao et al., 2006)

• Solving the two problems of Ranking SVM
• Higher weight on important grade pairs $\tau_{k(i)}$
• Normalization weight on pairs in query $\mu_{q(i)}$
• IR SVM = Ranking SVM using modified hinge loss
Modified Hinge Loss function

\[
\min_w \sum_{i=1}^l \tau_{k(i)} \mu_{q(i)} \left[ 1 - y_i \langle w, x_i^{(1)} - x_i^{(2)} \rangle \right]_+ + \lambda \| w \|^2
\]
Learning of IR SVM

\[
\min_w \sum_{i=1}^l \tau_{k(i)} \mu_{q(i)} \left[ 1 - y_i \langle w, x_i^{(1)} - x_i^{(2)} \rangle \right]_+ + \lambda \| w \|^2
\]

\[
\min_{w, \xi} \frac{1}{2} \| w \|^2 + \sum_{i=1}^l C_i \xi_i
\]

\[
y_i \langle w, x_i^{(1)} - x_i^{(2)} \rangle \geq 1 - \xi_i \quad i = 1, \ldots, l
\]

\[
\xi_i \geq 0
\]

\[
C_i = \frac{\tau_{k(i)} \mu_{q(i)}}{2 \lambda}
\]
AdaRank
Listwise Loss

$$\max_{f \in \mathcal{F}} \sum_{i=1}^{m} E(\pi(q_i, d_i, f), y_i)$$

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{m} (1 - E(\pi(q_i, d_i, f), y_i))$$
AdaRank
(Xu and Li, 2007)

• Optimizing exponential loss function
• Algorithm: AdaBoost-like algorithm for ranking
Loss Function of AdaRank

\[
\max_{f \in \mathcal{F}} \sum_{i=1}^{m} E(\pi(q_i, d_i, f), y_i)
\]

\[
\min_{f \in \mathcal{F}} \sum_{i=1}^{m} (1 - E(\pi(q_i, d_i, f), y_i))
\]

\[
e^{-x} \geq 1 - x
\]

\[
\min_{f \in \mathcal{F}} \sum_{i=1}^{m} \exp\{-E(\pi(q_i, d_i, f), y_i)\}
\]

\[
f(\tilde{x}) = \sum_{t=1}^{T} \alpha_t h_t(\tilde{x})
\]

\[
\min_{h_t \in \mathcal{H}, \alpha_t \in \mathbb{R}^+} L(h_t, \alpha_t) = \sum_{i=1}^{m} \exp\{-E(\pi(q_i, d_i, f_{t-1} + \alpha_t h_t), y_i)\}
\]

Any evaluation measure taking value between \([-1, +1]\)
AdaRank Algorithm

Input: \( S = \{(x_i, y_i)\}_{i=1}^{m} \)
Parameter: \( T \) (number of iterations)
Evaluation measure: \( E \)
Initialize \( P_1(i) = 1/m \)
For \( t = 1, \ldots, T \)
  \begin{itemize}
  \item Create weak ranker \( h_t \) with weighted distribution \( P_t \) on training data \( S \)
  \item Choose \( \alpha_t \)
    \[ \alpha_t = \frac{1}{2} \cdot \ln \frac{\sum_{i=1}^{m} P_t(i) (1 + E(\pi_i, y_i))}{\sum_{i=1}^{m} P_t(i) (1 - E(\pi_i, y_i))} \]
    where \( \pi_i = \text{sort}_{h_t}(x_i) \)
  \item Create \( f_t \)
    \[ f_t(x) = \sum_{k=1}^{t} \alpha_k h_k(x) \]
  \item Update \( P_{t+1} \)
    \[ P_{t+1}(i) = \frac{\exp(-E(\pi_i, y_i))}{\sum_{j=1}^{m} \exp(-E(\pi_j, y_j))} \]
    where \( \pi_i = \text{sort}_{f_t}(x_i) \)
  \end{itemize}
End For
Output: the ranking model \( f(x) = f_T(x) \)
3. *Learning to Match*
3-1. Semantic Matching in Search
Query Document Mismatch is Biggest Challenge in Search
Query Document Mismatch

• Same intent can be represented by different queries (representations)
• Search is still mainly based on term level matching
• Query document mismatch occurs, when searcher and author use different representations
## Examples of Query Document Mismatch

<table>
<thead>
<tr>
<th>Query</th>
<th>Document</th>
<th>Term Matching</th>
<th>Semantic Matching</th>
</tr>
</thead>
<tbody>
<tr>
<td>seattle best hotel</td>
<td>seattle best hotels</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>pool schedule</td>
<td>swimmingpool schedule</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>natural logarithm transformation</td>
<td>logarithm transformation</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>
Matching at Different Levels

Semantic Matching

Form
Phrase
Sense
Topic
Structure

Term Matching
Query Understanding

Structure Identification

Topic Identification

Similar Query Finding

Phrase Identification

Spelling Error Correction

main phrase: michael jordan

topic: machine learning, berkeley

similar query: michael i. jordan

phrase: michael jordan

phrase: berkeley

query form: michael jordan berkeley

Structure

Topic

Sense

Phrase

Term

michael jordan berkele
Title Structure Identification

Topic Identification

Key Phrase Identification

Phrase Identification

**main phrase in title:** michael jordan

**topic:** machine learning, berkeley

**key phrase:** michael jordan, professor, electrical engineering

**phrase:** michael jordan, professor, department, electrical engineering

Homepage of Michael Jordan

Michael Jordan is Professor in the Department of Electrical Engineering
Semantic Matching

Relevance Ranking

Query Representation → Query Document Matching → Document Representation

**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
Matching in Different Ways

- Query Reformulation
- Query and document transformation
- No transformation
- Document transformation
Web Search System

User Interface -> Query Understanding -> Retrieving

Ranking <- Query Document Matching

Crawling -> Document Understanding -> Indexing

Index
Machine Learning for Query Document Matching in Web Search
Learning for Matching between Query and Document

- Learning matching function
  \[ f_M(q,d) \text{ or } p_M(r | q,d) \]

- Using training data \((q_1,d_1,r_1), \ldots, (q_N,d_N,r_N)\)

- \(q_1,q_2,\ldots,q_N\) and \(d_1,d_2,\ldots,d_N\) can be id’s or feature vectors

- \(r_1,r_2,\ldots,r_N\) can be binary or numerical values

- *Using relations in data and/or prior knowledge*
Challenges in Machine Learning for Matching

• How to leverage relations in data
• How to handle complicated data (e.g., long queries, questions)
• How to scale up
• How to leverage prior knowledge
• How to deal with tail
Long Tail Challenge

- Head pages have rich anchor texts and click data
- Tail queries and pages suffer more from mismatch
- Problem of propagating information and knowledge from head to tail
Relation between Matching and Ranking

• In traditional IR:
  – Ranking = matching
    \[ f(q, d) = f_{BM25}(q, d) \text{ or } f(q, d) = P_{LMIR}(d \mid q) \]

• Web search:
  – Ranking and matching become separated
  – Learning to rank becomes state-of-the-art
    \[ f(q, d) = f_{BM25}(q, d) + g_{PageRank}(d) + \cdots \]
  – Matching = feature learning for ranking
Matching vs Ranking

In search, first matching and then ranking

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Matching</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Matching degree between query and document</td>
<td>Ranking list of documents</td>
</tr>
<tr>
<td>Model</td>
<td>f(q, d)</td>
<td>f(q,d1), f(q,d2), ... f(q,dn)</td>
</tr>
<tr>
<td>Challenge</td>
<td>Mismatch</td>
<td>Correct ranking on top</td>
</tr>
</tbody>
</table>
Matching Functions as Features in Learning to Rank

- Term level matching: \( f_{BM25}(q,d) \) \( f_{n-BM25}(q,d) \)
- Phrase level matching: \( f_P(q,d) \)
- Sense level matching: \( f_S(q,d) \)
- Topic level matching: \( f_T(q,d) \)
- Structure level matching: \( f_C(q,d) \)
- Term level matching (spelling, stemming): \( q' \rightarrow q \)
Approaches to Learning for 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
Semantic Matching in Search

Hang Li
Huawei Technologies, Hong Kong
hangli.hl@huawei.com

Jun Xu
Huawei Technologies, Hong Kong
nkxujun@gmail.com
3-2. Overview of Learning to Match
Matching between Heterogeneous Data is Everywhere

- Matching between user and product (collaborative filtering)
- Matching between text and image (image annotation)
- Matching between people (dating)
- Matching between languages (machine translation)
- Matching between receptor and ligand (drug design)
Formulation of Learning Problem

• Learning matching function

\[ f(x, y) \]

• Training data \((x_1, y_1, r_1), \ldots, (x_N, y_N, r_N)\)

• Generated according to

\[ x \sim P(X), \quad y \sim P(Y \mid X), \quad r \sim P(R \mid X, Y) \]
Formulation of Learning Problem

- Loss Function
  \[ L(r, f(x, y)) \]

- Risk Function
  \[ R(r, f(x, y)) = \int_{\mathbb{X} \times \mathbb{Y} \times \mathbb{R}} P(x, y, r) L(r, f(x, y)) dP(x, y, r) \]

- Objective Function in Learning
  \[ \min_{f \in \mathcal{F}} \sum_{i=1}^{N} L(r_i, f(x_i, y_i)) + \Omega(f) \]
### Matching Problem: Instance Matching

Instances can be represented as matching between nodes in a bipartite graph.

<table>
<thead>
<tr>
<th></th>
<th>y1</th>
<th>y2</th>
<th>y3</th>
<th>yn</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td></td>
<td></td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>x3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xm</td>
<td></td>
<td></td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>
### Matching Problem: Feature Matching

Features can be represented as matching between objects in two spaces.
Matching Problem: Structure Matching

<table>
<thead>
<tr>
<th>Structures</th>
<th>y1</th>
<th>y2</th>
<th>y3</th>
<th>yn</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>x2</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>x3</td>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>xm</td>
<td>1</td>
<td></td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
3-3. Methods of Learning to Match
Regularized Matching in Latent Space (RMLS)
Matching in Latent Space
(Wu et al., WSDM 2013, JMLR 2013)

• Motivation
  – Matching between query and document in latent space

• Assumption
  – Queries have similarity
  – Document 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
• Matching between Heterogeneous Data
• Example: Image Annotation
Projecting Keywords and Images into Latent Space
IR Models as Similarity Functions
(Xu and Li 2010)

Query Space

Document Space

New Space

Mapping functions are diagonal matrices

VSM, BM25, LM, MRF

unigram

unigram

unigram

unigram

unigram

unigram

unigram
IR Models Are Similarity Functions

- **VSM**
  - \( BM25(q, d) = \langle \phi^VSM_Q(q), \phi^VSM_D(d) \rangle \), for all \( w \in V \)
  - \( \phi^VSM_Q(q)_w = tfidf(w, q) \) and \( \phi^VSM_D(d)_w = tfidf(w, d) \)

- **BM25**
  - \( BM25(q, d) = \langle \phi^{BM25}_Q(q), \phi^{BM25}_D(d) \rangle \), for all \( w \in V \)
  - \( \phi^{BM25}_Q(q)_w = \frac{(k_3+1) \times tf(w, q)}{k_3 + tf(w, q)} \)
  - \( \phi^{BM25}_D(d)_w = IDF(w) \cdot \frac{(k_1+1) \times tf(w, d)}{k_1(1-b+b \cdot \frac{len(d)}{avgDocLen}) + tf(w, d)} \)

- **LMIR**
  - \( LMIR(q, d) = \langle \phi^{LMIR}_Q(q), \phi^{LMIR}_D(d) \rangle + \text{len}(q) \cdot \log \frac{\mu}{\text{len}(d)+\mu} \), for all \( w \in V \)
  - \( \phi^{LMIR}_Q(q)_w = tf(w, q) \)
  - \( \phi^{LMIR}_D(d)_w = \log \left( 1 + \frac{tf(w, d)}{\mu \cdot P(w)} \right) \), where \( P(w) \) plays similar role as IDF in BM25
Partial Least Square (PLS)

• Setting
  – Two spaces: \( q \in Q, \ d \in D \)

• Input
  – Training data: \( \{(q_i, d_i, c_i)\} \quad c_i \in N^+ \)

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

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

• Optimization

\[
\arg \max_{L_q, L_d} \sum_{(q_i, d_i)} c_i \langle L_q \cdot q_i, L_d \cdot d_i \rangle \quad \text{st.} \quad L_q^T L_q = I, \quad L_d^T L_d = I
\]
Solution of Partial Least Square

• Non-convex optimization
• Global optimal solution exists
• Global optimum can be found by solving SVD (Singular Value Decomposition)
Regularized Mapping to Latent Space (RMLLS)

- **Setting**
  - Two spaces: \( q \in Q, d \in D \)

- **Input**
  - Training data: \( \{(q_i, d_i, c_i)\} \quad c_i \in N^+ \)

- **Output**
  - Matching function \( f(q, d) \)

- **Assumption**
  - Two linear (and sparse) transformations \( L_q, L_d \)
  - Dot product as similarity function \( \langle L_q \cdot q, L_d \cdot d \rangle \)

- **Optimization**

\[
\arg \max_{L_q, L_d} \sum_{(q_i, d_i)} c_i \langle L_q \cdot q_i, L_d \cdot d_i \rangle \quad \text{st.} \quad |l_q| \leq \theta_q, |l_d| \leq \theta_d, \|l_q\| \leq \tau_q, \|l_d\| \leq \tau_d
\]
Solution of Regularized Mapping to Latent Space (RMLS)

• Coordinate Descent
• Repeat
  – Fix \( L_q \), update \( L_d \)
  – Fix \( L_d \), update \( L_q \)

• No guarantee to find global optimum
• Updates can be parallelized by rows
## Comparison between PLS and RMLS

<table>
<thead>
<tr>
<th></th>
<th>PLS</th>
<th>RMLS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Assumption</strong></td>
<td>Orthogonal</td>
<td>L1 and L2 Regularization</td>
</tr>
<tr>
<td><strong>Optimization Method</strong></td>
<td>Singular Value Decomposition</td>
<td>Coordinate Descent</td>
</tr>
<tr>
<td><strong>Optimality</strong></td>
<td>Global optimum</td>
<td>Local optimum</td>
</tr>
<tr>
<td><strong>Efficiency</strong></td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>
String Re-writing Kernel
Learning for String Re-wr-ing

((distance between sun and earth), (how far sun earth), +1)
((distance between beijing and shanghai), (how far is beijing from shanghai), +1)
... ....
((distance between moon and earth), (how far sun earth), -1)

+1/-1?

Learning System

Prediction System

Model

((distance between moon and earth), (how far is earth from moon))
String Re-writing Kernel
(Bu et al., 2012)

• Training data

\[((s_1, t_1), y_1) \cdots ((s_n, t_n), y_n)\]

• Model

\[y = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K((s_i, t_i), (s, t))\right) \quad \alpha_i \geq 0\]

• String Re-writing Kernel

\[K((s_i, t_i), (s, t))\]
Re-Writing Rule

- Rule is applied to re-writing of substrings

Shakespeare wrote Hamlet in 16 century

Hamlet was written by Shakespeare

Cao Xueqin wrote Dream of the Red Chamber

Dream of the Red Chamber was written by Cao Xueqin

Cao Xueqin wrote Dream of the Red Chamber

Hamlet was written by Shakespeare

Re-writing Rule
Re-Writing Rule (2)

• Multiple re-writing rules may be applied

Shakespeare wrote Hamlet in 16 century

Hamlet was written by Shakespeare

Re-writing Rule

* wrote * in 16 century

* was written by *
Definition of String Re-writing Kernel (SRK)

\[ K\left((s_1, t_1), (s_2, t_2)\right) = \left\langle \Phi(s_1, t_1), \Phi(s_2, t_2) \right\rangle \]

\[ \Phi(s, t) = (\phi_r(s, t))_{r \in R} \]

\[ \phi_r(s, t) = n\lambda^i \quad \lambda \in (0, 1] \]

\[ K\left((s_1, t_1), (s_2, t_2)\right) = \sum_r \phi_r(s_1, t_1)\phi_r(s_2, t_2) \]
Similarity between Two Re-writings

- Similarity between two re-writings with respect to re-writing rules

Shakespeare wrote *Hamlet* in 16 century

Hamlet was written by Shakespeare

Cao Xueqin wrote *Dream of the Red Chamber*

Dream of the Red Camber was written by Cao Xueqin

$$\langle r_1 = 0, \ldots, r_{100} = 1, \ldots, r_{1001} = 1, \ldots \rangle$$

$$\langle r_1 = 0, \ldots, r_{100} = 1, \ldots, r_{1001} = 0, \ldots \rangle$$
A Sub-Class of String Re-writing Kernel

- Advantage: Matching between informally written sentences such as long queries in search can be effectively performed
- Challenge
  - Number of re-writing rules is infinite
  - Number of applicable rules increase exponentially when length of sentence increases
- Our Approach
  - Sub-class: kb-SRK
Definition of kb-SRK

• Special class of SRK

• Re-writing rules in kb-SRK
  – String patterns in rule are of length k
  – Wildcard ? only substitutes a single character
  – Alignment between string patterns is bijective

? wrote ?
  ? described ?
Reformulation of kb-SRK

Definition

\[ K((s_1, t_1), (s_2, t_2)) = \sum_{r} \phi_r(s_1, t_1) \phi_r(s_2, t_2) \]

\[ \phi_r(s, t) = \sum_{\alpha_s \in k-gram(s)} \sum_{\alpha_t \in k-gram(t)} \bar{\phi}_r(\alpha_s, \alpha_t) \]

\[ \bar{\phi}_r(\alpha_s, \alpha_t) = \lambda^i \text{, if } \alpha_s, \alpha_t \text{ match, } \bar{\phi}_r(\alpha_s, \alpha_t) = 0, \text{ otherwise} \]

Equivalent formulation based on fact that summations are exchangable

\[ K((s_1, t_1), (s_2, t_2)) = \sum_{\alpha_{s_1} \in k-gram(s_1)} \sum_{\alpha_{t_1} \in k-gram(t_1)} \sum_{\alpha_{s_2} \in k-gram(s_2)} \sum_{\alpha_{t_2} \in k-gram(t_2)} \bar{K}_k((\alpha_{s_1}, \alpha_{t_1}), (\alpha_{s_2}, \alpha_{t_2})) \]

\[ \bar{K}_k = \sum_{r \in R} \bar{\phi}_r(\alpha_{s_1}, \alpha_{t_1}) \bar{\phi}_r(\alpha_{s_2}, \alpha_{t_2}) \]
Similarity between Re-writings of K-gram Pairs

two k-gram pairs

\[ \alpha_{s_1} = abbcbbb, \quad \alpha_{s_2} = abccccdd \]

\[ \alpha_{t_1} = cbcbcbcb, \quad \alpha_{t_2} = cbccdcdd \]

\[ r_{200} \]

a b ? c c ?

applicable to both pairs

c b c ? ? c ?

\[ \langle r_1 = 0, \ldots, r_{200} = 1, \ldots, r_{2001} = 1, \ldots \rangle \]

\[ r_{2001} \]

a b ? c c ?

not applicable to second pair

c ? c b ? c ?

\[ \langle r_1 = 0, \ldots, r_{200} = 1, \ldots, r_{2001} = 0, \ldots \rangle \]

Only need to consider the rules applicable to both k-gram pairs
Efficiently Calculating Number of Applicable Rules on Combined k-gram Pairs

two k-gram pairs

\[ \alpha_{s_1} = abbc\text{cb}bb \quad \alpha_{s_2} = abcc\text{cd}d \]

\[ \alpha_{t_1} = cbcb\text{bb}cb \quad \alpha_{t_2} = cbcc\text{d}cd \]

applicable rule

\[ ab?cc?? \]
\[ cbc??c? \]

combined k-gram pairs

\[ \alpha_s = (a, a)(b, b)(b, c)(c, c)(c, c)(b, d)(b, d) \]
\[ \alpha_t = (c, c)(b, b)(c, c)(b, c)(b, d)(c, c)(b, d) \]

applicable rule

\[ (a, a)(b, b) ?(c, c)(c, c) ?? \]
\[ (c, c)(b, b)(c, c) ??(c, c) ? \]

identical doubles can be either or not substituted
non-identical doubles must be substituted

\[ \overline{K}_k = (1)(1 + \lambda^2)(\lambda^2)(2! \lambda^4)(1 + 6 \lambda^2 + 6 \lambda^4) \]
Time Complexity of Calculation of kb-SRK

• Inner loop
  – Time complexity $O(k)$

• Outer loop
  – Empirical time complexity $O(l^2 \cdot k)$

$$l = \max(|s_1|, |t_1|, |s_2|, |t_2|)$$
4. Applications to Social Media
4.1 Tweet Recommendation
Tweet Recommendation

<table>
<thead>
<tr>
<th>users</th>
<th>i1</th>
<th>i2</th>
<th>i3</th>
<th>in</th>
</tr>
</thead>
<tbody>
<tr>
<td>u1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>u2</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>u3</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>um</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SVD Feature: Feature-based Matrix Factorization (Chen et al., 2012)

• Model

\[ r(u,i) = \langle l_g, g \rangle + \langle l_u, u \rangle + \langle l_g, i \rangle + \langle L_u u, L_i i \rangle \]

• Loss function: squared loss

• Algorithm: Gradient descent
Matching between User and Item

Matching in latent space

\[ L_u u \times L_i i \]

- \( u \) user vector
- \( i \) item vector
- \( g \) global feature vector
4.2 Short Text Conversation
Short Text Conversation

One Step toward Turing Test
Important subtask of dialogue

Human: a message
Computer: a reply
Massive Amount of Data Available

Our paper entitled learning to rank has been accepted by ACL.

We are lucky. Our paper has been accepted by SIGIR this year. We are going to present it.

The PC of WSDM noticed us that our paper has been accepted.
System of Retrieval-based Short Text Conversation

- Given post, find most suitable response
- Large repository of post-response pairs
- Take it as search problem

![Diagram of the System of Retrieval-based Short Text Conversation]

- Short text
- Retrieval
- Retrieved posts and responses
- Matching
- Matched responses
- Ranking
- Ranked responses
- Best response

Index of post and response
Learning to match
Learning to rank
Learning to Match for Short Text Conversation

$t_1$ $c_1$
$t_2$ $c_2$
$...$
$t_n$ $c_n$
$t(n+1)$

Learning System

Model

Matching System

$c(n+1)$
Linear Matching Model (RMLS)
(Wang et al. EMNLP 2013)

- Short text = sum of word vectors
- Matching degree between short texts = Inner product of their document vectors
- Word vectors are learned from short text data
Deep Matching Model (Lu & Li, NIPS 2013)

• Take interactions between texts as input
• Learning features in different granularities using LDA
• Learning parameters using back-propagation
Representing Posts and Comments as Bags of Words

Words in post and comment are viewed as different words
Constructing Topics Using Latent Dirichlet Allocation

Bag of words

- P_mats, P_catch, P_mice, C_great, C_mats
- P_mats, P_chase, P_mice, C_poor, C_mice

Topic

- P_mats, P_mice, C_mice, C_mats
- P_chase, P_catch, C_great, C_poor, ....

Topics in different granularities form different hidden layers
Architecture of Deep Matching Model

Examples
Local Model 1: (特产, 土产, 味道,...) || (豆腐, 烤鸭, 甜, 野味, 糯米...)
Local Model 2: (路程, 安排, 地点,...) || (距离, 安全, 隧道, 高速, 机票...)

烤鸭啊，想吃热乎的去烤鸭店，如全聚德，真空包装的超市就有
5. Summary
Summary

• Learning to rank and learning to match are key technologies for many applications including search and recommendation

• Learning to Rank
  – Ranking SVM
  – IR SVM
  – AdaRank

• Learning to Match
  – Regularized Matching in Semantic Space
  – String Re-writing Kernel

• Many applications on social media can be constructed with learning to rank and learning to match techniques
References


• Hang Li, A Short Introduction to Learning to Rank, IEICE Transactions on Information and Systems, E94-D(10), 2011.


Thank You!

hangli.hl@huawei.com