Learning to Rank: From Pairwise Approach to Listwise Approach

Hang Li
Microsoft Research Asia

Joint work with Tie-Yan Liu, Jun Xu, and others
Talk Outline

• Ranking in Information Retrieval
• Learning to Rank
• ListNet: Probabilistic Model for Ranking
• AdaRank: Boosting Algorithm for Ranking
Means for Information Access

Information Extraction

Summarization

Information Retrieval
Current Approach
Information Retrieval = Ranking
Ranking Problem: Example = Document Retrieval

documents

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

query \(q\)

Ranking System

ranking of documents

\[ d_{q,1}, d_{q,2}, \ldots, d_{q,n_q} \]
Ranking based on Relevance, Importance, Preference
Learning to Rank
Issues of Learning to Rank

• General Framework
• Features
• Data Labeling
• Evaluation Measures
• Characteristics
• Existing Approaches
Learning to Rank

$q_1 \quad q_m$
$d_{1,1} \quad d_{m,1}$
$d_{1,2} \quad d_{m,2}$
$\vdots \quad \vdots$
$d_{1,n_1} \quad d_{m,n_m}$

Learning System

Model

$q_{m+1}$

Ranking System

$d_{m+1,1}$
$d_{m+1,2}$
$\vdots$
$d_{m+1,n_{m+1}}$
Score based Ranking

\[ q_1, q_2, \ldots, q_m, d_{1,1}, d_{1,2}, \ldots, d_{1,n_1}, \ldots, d_{m,1}, d_{m,2}, \ldots, d_{m,n_m}, q_{m+1} \]

Learning System

\[ f(q, d) \]

Ranking System

\[ d_{m+1,1} f(q_{m+1}, d_{m+1,1}), d_{m+1,2} f(q_{m+1}, d_{m+1,2}), \ldots, d_{m+1,n_{m+1}} f(q_{m+1}, d_{m+1,n_{m+1}}) \]
Learning Process

\[
\begin{align*}
\{d_{1,1} & \quad y_{1,1} & \quad \{x_{1,1} & \quad y_{1,1} \\
\vdots & \quad \vdots & \quad \vdots & \quad \vdots \\
d_{1,n_1} & \quad y_{1,n_1} & \quad x_{1,n_1} & \quad y_{1,n_1} \}
\end{align*}
\]

Data Labeling

\[
\begin{align*}
\{d_{m,1} & \quad y_{m,1} & \quad \{x_{m,1} & \quad y_{m,1} \\
\vdots & \quad \vdots & \quad \vdots & \quad \vdots \\
d_{m,n_m} & \quad y_{m,n_m} & \quad x_{m,n_m} & \quad y_{m,n_m} \}
\end{align*}
\]

Feature Extraction

Learning \quad y = f(x)
Features

• Relevance: BM25
• Importance: PageRank
Relevance

- No rigorous definition
- Query = “soccer”, document = about soccer → document relevant
- Judgment by humans: several levels, e.g. “definitely relevant”, “partially relevant”
Key for Relevance: Matching between Query and Document

\[ f(q, d) \]
Importance

• No rigorous definition
• Citation, quality, freshness
• Independent of query
Key for Importance on Web Data: Link between Documents

\[ f(\cdot, d) \]
Data Labeling Methods

• Rank
  – e.g., relevant, partially relevant, irrelevant
• Ordered Pair
  – e.g., A > B, B > C
• Others are impractical
  – List
  – Score
Evaluation Measures

- NDCG (Normalized Discounted Cumulative Gain)
- MAP (Mean Average Precision)
- MRR (Mean Reciprocal Rank)
- WTA (Winners Take All)
NDCG

- Example: perfect ranking for $q_i$
  - $(3, 3, 2, 2, 1, 1, 1)$ rank $r = 3, 2, 1$
  - $(7, 7, 3, 3, 1, 1, 1)$ gain $2^{r(j)} - 1$
  - $(1, 0.63, 0.5, 0.43, 0.39, 0.36, 0.33)$ position discount
  - $(7, 18.11, 24.11, ...)$ DCG
    \[
    \sum_{j=1}^{m} \left( \frac{2^{r(j)} - 1}{\log(1 + j)} \right)
    \]
  - $(1/7, 1/18.11, 1/24.11, ...)$ normalizing factor $n_i$
  - $(1, 1, 1,1,1,1,1)$ 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, 14.11, 20.11, … )  DCG
  – (0.43, 0.78, 0.83, …. )  NDCG

• An imperfect ranking for the query will always decrease NDCG
Characteristics of Learning to Rank

• No need to predict category (or ordered category) vs Classification
• No need to predict value of $f(q,d)$ vs Regression
• Ranking order is more important vs Ordinal regression
Ordinal Regression
(Ordinal Classification)

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

• Prediction
  – Map to ordered categories
Learning to Rank for Information Retrieval

• Important to ranking top results correctly
• Numbers of documents vary according to queries
Existing Approaches

• Pairwise Approach
  – Ranking SVM (Herbrich et al 2000)
  – RankBoost (Freund et al 2003)
  – Ranknet (Burges et al 2005)

• Listwise Approach
  – ListNet (Cao et al 2007)
  – AdaRank (Xu & Li 2007)
  – SVM Method for Maximizing MAP (Yue et al 2007)
Previous Work: Pairwise Approach

- Transforming ranked list into document pairs
- Formalizing ranking as classification on document pairs
- Ranking SVM, RankNet, RankBoost

\[
\begin{align*}
q^{(i)} \\
\left( d^{(i)}_1, 5 \right) \\
\left( d^{(i)}_2, 3 \right) \\
\vdots \\
\left( d^{(i)}_{n(i)}, 2 \right)
\end{align*}
\]

Transform

\[
\left\{ \left( d^{(i)}_1, d^{(i)}_2 \right), \left( d^{(i)}_1, d^{(i)}_{n(i)} \right), \ldots, \left( d^{(i)}_2, d^{(i)}_{n(i)} \right) \right\}
\]

\[
\begin{align*}
5 & > 3 \\
5 & > 2 \\
3 & > 2
\end{align*}
\]
Problems with Pairwise Approach

- Loss function is suboptimal
  - Not to optimize evaluation measures (e.g. NDCG and MAP)
  - Not to consider position in ranking and number of documents per query

Pairwise loss vs. (1-NDCG@5)
TREC Dataset
Our Proposal: Listwise Approach
ListNet: Probabilistic Model for Ranking
Distance between Ranked Lists

• A Simple Example:
  – function $f$: $f(A)=3, f(B)=1, f(C)=0$  
  – function $h$: $h(A)=5, h(B)=2, h(C)=3$  
  – ground truth $g$: $g(A)=5, g(B)=3, g(C)=2$  

• Question: which function is closer to ground truth?

• Our proposal:
  – Ranked list $\leftrightarrow$ Permutation probability distribution
Permutation Probability

- Probability of permutation $\pi$ is defined as

$$P_s(\pi) = \prod_{j=1}^{n} \frac{\phi(s_{\pi(j)})}{\sum_{k=j}^{n} \phi(s_{\pi(k)})}$$

- Example:

$$P_f(\text{ABC}) = \frac{\phi(f(A))}{\phi(f(A)) + \phi(f(B)) + \phi(f(A)))} \cdot \frac{\phi(f(B))}{\phi(f(B)) + \phi(f(C)) + \phi(f(B))} \cdot \frac{\phi(f(C))}{\phi(f(C))}$$
Properties of Permutation Probability

• Function $f$: $f(A)=3$, $f(B)=1$, $f(C)=0$ \hspace{1cm} ABC
• Property 1: $P(ABC)$ is largest, $P(CBA)$ is smallest
• Property 2: swap B and C in ABC, $P(ABC) > P(ACB)$
Top-k Probability

- Computation of Permutation Probability is intractable
- Top-k Probability
  - Defining Top-k subgroup $G(j_1, \ldots, j_k)$ containing all permutations whose top-k documents are $j_1, \ldots, j_k$
  
  \[ P_s(G(j_1, \ldots, j_k)) = \prod_{t=1}^{k} \frac{\varphi(s_{\pi(j_t)})}{\sum_{u=t}^{n} \varphi(s_{\pi(j_u)})} \]

  - Time complexity of computation: from $n!$ to $n!/(n-k)!$
ListNet Method (*ICML’07*)

- Loss function = KL-divergence between two Top-$k$ probability distributions ($\varphi = \exp$)

\[
L(w) \propto - \sum_{q \in Q} \sum G(j_1, \ldots, j_k) \left( \prod_{t=1}^{k} \frac{\exp(s_{y(j_t)})}{\sum_{u=t}^{n} \exp(s_{y(j_u)})} \right) \log \left( \prod_{t=1}^{k} \frac{\exp(w \cdot X_{y(j_t)})}{\sum_{u=t}^{n} \exp(w \cdot X_{y(j_u)})} \right)
\]

- Model = Neural Network
- Algorithm = Gradient Descent
Experimental Results

Pairwise (RankNet)  Listwise (ListNet)

Training Performance on TREC Dataset
Experimental Results

NDCG@
Testing Performance on TREC Dataset
AdaRank: Boosting Algorithm for Ranking
Traditional Pairwise Approach (Training)

1. Generating document pairs
2. Document pairs
3. Minimizing # inversed document pairs

Minimizing loss function based on document pairs

Diagram:
- Query: $q_1, q_2, \ldots, q_m$
- Documents: $d_{1,1}, y_{1,1}, d_{1,2}, y_{1,2}, \ldots, d_{1,n_1}, y_{1,n_1}, \ldots, d_{m,1}, y_{m,1}, d_{m,2}, y_{m,2}, \ldots, d_{m,n_m}, y_{m,n_m}$
- Learning System
- Ranking model
Traditional Pairwise Approach (Evaluation)

\[
q_{m+1} \rightarrow \text{Ranking System} \rightarrow \text{Ranking model} \rightarrow d_{m+1,1}, y_{m+1,1}
\]

\[
d_{m+1,2}, y_{m+1,1} \\
\vdots \\
d_{m+1,n_{m+1}}, y_{m+1,n_{m+1}}
\]

IR measure (MAP, NDCG, ...)

Listwise performance measure
Loss Function of AdaRank

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

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

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

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

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

\[
f(\vec{x}) = \sum_{i=1}^{T} \alpha_i h_i(\vec{x})
\]

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

11/16/2007
AdaRank Algorithm

Input: $S = \{(q_i, d_i, y_i)\}_{i=1}^m$, and parameters $E$ and $T$

Initialize $P_1(i) = 1/m$.

For $t = 1, \ldots, T$

- Create weak ranker $h_t$ with weighted distribution $P_t$ on training data $S$.
- Choose $\alpha_t$
  \[ \alpha_t = \frac{1}{2} \cdot \ln \frac{\sum_{i=1}^m P_t(i)\{1 + E(\pi(q_i, d_i, h_t), y_i)\}}{\sum_{i=1}^m P_t(i)\{1 - E(\pi(q_i, d_i, h_t), y_i)\}}. \]
- Create $f_t$
  \[ f_t(\vec{x}) = \sum_{k=1}^t \alpha_k h_k(\vec{x}). \]
- Update $P_{t+1}$
  \[ P_{t+1}(i) = \frac{\exp\{-E(\pi(q_i, d_i, f_t), y_i)\}}{\sum_{j=1}^m \exp\{-E(\pi(q_j, d_j, f_t), y_j)\}}. \]

End For

Output ranking model: $f(\vec{x}) = f_T(\vec{x})$. 
Theoretical Results on AdaRank

• Training error will be continuously reduced during learning phase.

Theorem 1. The following bound holds on the ranking accuracy of the AdaRank algorithm on training data:

\[ \frac{1}{m} \sum_{i=1}^{m} E(\pi(q_i, d_i, f_{t}), y_i) \geq 1 - \prod_{t=1}^{T} e^{-\delta_{\text{min}}} \sqrt{1 - \varphi(t)^2}, \]

where \( \varphi(t) = \sum_{i=1}^{m} P_t(i) E(\pi(q_i, d_i, h_t), y_i) \), \( \delta_{\text{min}}^t = \min_{i=1,\ldots,m} \delta_i^t \), and

\[ \delta_i^t = E(\pi(q_i, d_i, f_{t-1} + \alpha_i h_t), y_i) - E(\pi(q_i, d_i, f_{t-1}), y_i) \]

\[ -\alpha_t E(\pi(q_i, d_i, h_t), y_i), \]

for all \( i = 1, 2, \cdots, m \) and \( t = 1, 2, \cdots, T \).
Ranking Accuracies on OHSUMED Data

![Bar chart showing ranking accuracies on OHSUMED Data for different metrics and algorithms. The chart includes metrics such as MAP, NDCG@1, NDCG@3, NDCG@5, and NDCG@10 for algorithms like BM25, Ranking SVM, RankBoost, AdaRank.MAP, and AdaRank.NDCG.](chart.png)
Ranking Accuracies on .Gov Data

- BM25
- Ranking SVM
- RankBoost
- AdaRank.MAP
- AdaRank.NDCG
Summary
Summary

• Ranking in Information Retrieval
• Learning to Rank
• ListNet: Probabilistic Model for Ranking
• AdaRank: Boosting Algorithm for Ranking
References

Thanks!