Learning a Deep Listwise Context Model for Ranking Refinement

Qingyao Ai¹, Keping Bi¹, Jiafeng Guo², W. Bruce Croft¹

¹CICS, University of Massachusetts Amherst
²ICT, Chinese Academy of Sciences

aiqy@cs.umass.edu
Outline

• Motivation
  – Query-specific learning to rank
  – Local context

• Our Approach
  – Learning to rank with local context
  – Deep Listwise Context Model
  – Attention Rank

• Experiments
  – Ranking performance
  – Pairwise ranking analysis

• Summary
Learning to Rank

• Given a query:

$$f(x)$$

Documents $\rightarrow$ Features $\rightarrow$ Ranking Scores

- $[0.1, 0.3, ..., -0.2]$
- $[-0.3, 0.5, ..., -0.8]$
- $[0.4, 0.2, ..., -0.1]$

- $3.2$
- $-1.4$
- $2.8$

Global feature assumption
Previous Studies

• Query-specific learning to rank:

Documents → Features → Ranking Scores

- Query features
  - Provide query information for the ranking model.
- Feature normalization
  - Twist input feature distributions to make them easier to use.

• Expensive
• Not adaptive
• Limited ability
Previous Studies

- Query-specific learning to rank:

  Documents → Features → Ranking Scores

  [0.1, 0.3, ..., -0.2]
  [-0.3, 0.5, ..., -0.8]
  [0.4, 0.2, ..., -0.1]

  3.2
  -1.4
  2.8

- Memory inefficient
- Data sparsity

- Query-specific ranking models
  - Build different rankers for different queries.

Previous Studies

• Learning to rank with local context:

Documents $\rightarrow$ Features $\rightarrow$ Ranking Scores

[0.1, 0.3, ..., -0.2]
[-0.3, 0.5, ..., -0.8]
[0.4, 0.2, ..., -0.1]

3.2 2.8

• Powerful and effective
• Memory efficient

Previous Studies

- Learning to rank with local context:

  Documents \[ \Rightarrow \] Features

  \[
  [0.1, 0.3, ..., -0.2] \\
  [-0.3, 0.5, ..., -0.8] \\
  [0.4, 0.2, ..., -0.1]
  \]

  \[ \Rightarrow \] Ranking Scores

  \[
  3.2 \\
  -1.4 \\
  2.8
  \]

  Documents \[ \Rightarrow \] Features

  \[
  [0.1, 0.3, ..., -0.2] \\
  [-0.3, 0.5, ..., -0.8] \\
  [0.4, 0.2, ..., -0.1]
  \]

  \[ \Rightarrow \] Ranking Scores

  \[
  3.2 \\
  -1.4 \\
  2.8
  \]

- Only work for text data
- Additional feature design/extraction

Research Questions

How to design a LTR model with local context:

1. Work for all types of LTR features.
Learning to Rank with Local Context

- Problem reformulation:

$$L = \sum_{q \in Q} \ell \left( \left\{ y_{(q,d)} \right\}, \left\{ \text{Score}_{(q,d)} \left| d \in D \right\} \right)$$

$$X_q = \left\{ \vec{x}_{(q,d)} \left| d \in R_q \right\} \right.$$
Deep Listwise Context Model (DLCM)

Input Document Vectors

Encoding Listwise Local Context

Re-ranking with Local Context

\[ R_q^n \]

\[ x(q,d_1) \] \[ o_n \] \[ \text{Score}(d_1) \]

\[ x(q,d_2) \] \[ o_{n-1} \] \[ \text{Score}(d_2) \]

\[ x(q,d_3) \] \[ o_{n-2} \] \[ \text{Score}(d_3) \]

... \[ ... \]

\[ x(q,d_n) \] \[ o_1 \] \[ \text{Score}(d_n) \]

GRU

\[ I(R_q^n, X_q^n) \] \[ s_n \] \[ \phi(o, s_n) \]
Feature abstraction and dimensionality increase: [Cheng et al. 2016]

\[ x'_{(q,d)} \]

*Input document vector*

\[ x_{(q,d)} \]

*Original LTR feature vector*

Why?
- Flexibility
- Robustness
Encoding Listwise Local Context

Sequentially encode the feature vectors of top results with RNN:

Why?

- Feature interaction
- Inherent structure
Re-ranking with Local Context

- Score documents with an attention function:

\[
\phi(\tilde{o}_{n+1-i}, \tilde{s}_n) = \tilde{V}_\phi \cdot (\tilde{o}_{n+1-i} \cdot \tanh(\tilde{W}_\phi \cdot \tilde{s}_n + \tilde{b}_\phi))
\]
Attention Rank

- Treat ranking as a problem of attention allocation:

\[ [0, 4, 0, 1] \]

\[ [1.2, 3.4, -0.3, 2.1] \]

Cross Entropy
Experimental Setup

• Dataset
  – Microsoft 30K
  – Microsoft 10K
  – Yahoo! Webscope v2.0

• Evaluation
  – Select the model with lowest loss on the validation set
  – Metrics: nDCG, ERR [Chapelle et al. 2009] (@1, @3, @5, @10)

<table>
<thead>
<tr>
<th></th>
<th>Queries</th>
<th>Doc.</th>
<th>Rel.</th>
<th>Feat.</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft 30K</td>
<td>31,531</td>
<td>3,771k</td>
<td>5</td>
<td>136</td>
<td>2010</td>
</tr>
<tr>
<td>Microsoft 10K</td>
<td>10,000</td>
<td>1,200k</td>
<td>5</td>
<td>136</td>
<td>2010</td>
</tr>
<tr>
<td>Yahoo! set 1</td>
<td>29,921</td>
<td>710k</td>
<td>5</td>
<td>700</td>
<td>2010</td>
</tr>
</tbody>
</table>
Model Setup

- **Initial Learning-to-rank system**
  - **SVMrank** [Joachims 2006]
    - SVM-based ranking model with pairwise loss
  - **LambdaMART** [Burges 2010]
    - The state-of-the-art tree model with listwise loss
Model Setup

Features

\[ \{ x_{q,d_1}, x_{q,d_2}, x_{q,d_3} \} \]

LTR system

Initial List

\[ \{ d_1, d_2, d_3 \} \]

Re-ranking model

Final List

\[ \{ d_2, d_1, d_3 \} \]

Loss Function

- Re-ranking model

DNN

Listwise Input DNN (LIDNN)

DLCM

\[ x_{q,d_1} \]

\[ \{ x_{q,d_1}, x_{q,d_3}, x_{q,d_2} \} \]

\[ \{ x_{q,d_1}, x_{q,d_3}, x_{q,d_2} \} \]
Model Setup

- **Features**
  \[
  \begin{align*}
  x_{q,d_1} \\
  x_{q,d_2} \\
  x_{q,d_3}
  \end{align*}
  \]

- **Initial List**
  \[
  \begin{align*}
  d_1 \\
  d_2 \\
  d_3
  \end{align*}
  \]

- **LTR system**

- **Re-ranking model**

- **Final List**
  \[
  \begin{align*}
  d_1 \\
  d_2 \\
  d_3
  \end{align*}
  \]

- **Features**
  - **Initial List**
  - **Final List**

- **Loss Function**
  - **ListMLE** [Xia et al. 2008]
    - Optimize the likelihood of the best ranking
  - **SoftRank** [Taylor et al. 2008]
    - Optimize nDCG directly
  - **Attention Rank (AttRank)**
    - Optimize listwise attention allocation
# Ranking Performance on SVMrank

* Microsoft 30K

<table>
<thead>
<tr>
<th>Model</th>
<th>Loss Func</th>
<th>NDCG@1</th>
<th>ERR@1</th>
<th>NDCG@5</th>
<th>ERR@5</th>
<th>NDCG@10</th>
<th>ERR@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>ListMLE</td>
<td>0.337</td>
<td>0.149</td>
<td>0.356</td>
<td>0.249</td>
<td>0.382</td>
<td>0.271</td>
</tr>
<tr>
<td></td>
<td>SoftRank</td>
<td>0.388</td>
<td>0.208</td>
<td>0.379</td>
<td>0.300</td>
<td>0.395</td>
<td>0.318</td>
</tr>
<tr>
<td></td>
<td>AttRank</td>
<td>0.395</td>
<td>0.198</td>
<td>0.396</td>
<td>0.297</td>
<td>0.415</td>
<td>0.316</td>
</tr>
<tr>
<td>LIDNN</td>
<td>ListMLE</td>
<td>0.291</td>
<td>0.122</td>
<td>0.331</td>
<td>0.222</td>
<td>0.362</td>
<td>0.245</td>
</tr>
<tr>
<td></td>
<td>SoftRank</td>
<td>0.315</td>
<td>0.141</td>
<td>0.341</td>
<td>0.238</td>
<td>0.367</td>
<td>0.260</td>
</tr>
<tr>
<td></td>
<td>AttRank</td>
<td>0.306</td>
<td>0.135</td>
<td>0.331</td>
<td>0.231</td>
<td>0.361</td>
<td>0.253</td>
</tr>
<tr>
<td>DLCM</td>
<td>ListMLE</td>
<td>0.339</td>
<td>0.149</td>
<td>0.357</td>
<td>0.248</td>
<td>0.381</td>
<td>0.269</td>
</tr>
<tr>
<td></td>
<td><strong>SoftRank</strong></td>
<td><strong>0.424</strong></td>
<td><strong>0.224</strong></td>
<td><strong>0.408</strong></td>
<td><strong>0.316</strong></td>
<td><strong>0.423</strong></td>
<td><strong>0.334</strong></td>
</tr>
<tr>
<td></td>
<td>AttRank</td>
<td>0.407</td>
<td>0.206</td>
<td>0.404</td>
<td>0.303</td>
<td>0.422</td>
<td>0.322</td>
</tr>
<tr>
<td></td>
<td>SVMrank</td>
<td>0.301</td>
<td>0.124</td>
<td>0.335</td>
<td>0.223</td>
<td>0.365</td>
<td>0.246</td>
</tr>
</tbody>
</table>
# Ranking Performance on SVMrank

<table>
<thead>
<tr>
<th>Model</th>
<th>Loss Func</th>
<th>NDCG@1</th>
<th>ERR@1</th>
<th>NDCG@5</th>
<th>ERR@5</th>
<th>NDCG@10</th>
<th>ERR@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>ListMLE</td>
<td>0.337</td>
<td>0.149</td>
<td>0.356</td>
<td>0.249</td>
<td>0.382</td>
<td>0.271</td>
</tr>
<tr>
<td></td>
<td>SoftRank</td>
<td>0.388</td>
<td>0.208</td>
<td>0.379</td>
<td>0.300</td>
<td>0.395</td>
<td>0.318</td>
</tr>
<tr>
<td></td>
<td>AttRank</td>
<td>0.395</td>
<td>0.198</td>
<td>0.396</td>
<td>0.297</td>
<td>0.415</td>
<td>0.316</td>
</tr>
<tr>
<td>LIDNN</td>
<td>ListMLE</td>
<td>0.291</td>
<td>0.122</td>
<td>0.331</td>
<td>0.222</td>
<td>0.362</td>
<td>0.245</td>
</tr>
<tr>
<td></td>
<td>SoftRank</td>
<td>0.315</td>
<td>0.141</td>
<td>0.341</td>
<td>0.238</td>
<td>0.367</td>
<td>0.260</td>
</tr>
<tr>
<td></td>
<td>AttRank</td>
<td>0.306</td>
<td>0.135</td>
<td>0.331</td>
<td>0.231</td>
<td>0.361</td>
<td>0.253</td>
</tr>
<tr>
<td>DLCM</td>
<td>ListMLE</td>
<td>0.339</td>
<td>0.149</td>
<td>0.357</td>
<td>0.248</td>
<td>0.381</td>
<td>0.269</td>
</tr>
<tr>
<td></td>
<td>SoftRank</td>
<td>0.424</td>
<td>0.224</td>
<td>0.408</td>
<td>0.316</td>
<td>0.423</td>
<td>0.334</td>
</tr>
<tr>
<td></td>
<td>AttRank</td>
<td>0.407</td>
<td>0.206</td>
<td>0.404</td>
<td>0.303</td>
<td>0.422</td>
<td>0.322</td>
</tr>
<tr>
<td>SVMrank</td>
<td></td>
<td>0.301</td>
<td>0.124</td>
<td>0.335</td>
<td>0.223</td>
<td>0.365</td>
<td>0.246</td>
</tr>
</tbody>
</table>

- **Observation**
  - All re-ranking models achieved improvements.
  - DLCM significantly outperformed our baselines.

*Microsoft 30K*
## Ranking Performance on LambdaMART

* Microsoft 30K

<table>
<thead>
<tr>
<th>Model</th>
<th>Loss Func</th>
<th>NDCG@1</th>
<th>ERR@1</th>
<th>NDCG@5</th>
<th>ERR@5</th>
<th>NDCG@10</th>
<th>ERR@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>ListMLE</td>
<td>0.372</td>
<td>0.174</td>
<td>0.386</td>
<td>0.278</td>
<td>0.409</td>
<td>0.299</td>
</tr>
<tr>
<td></td>
<td>SoftRank</td>
<td>0.384</td>
<td>0.209</td>
<td>0.378</td>
<td>0.302</td>
<td>0.397</td>
<td>0.321</td>
</tr>
<tr>
<td></td>
<td>AttRank</td>
<td>0.388</td>
<td>0.199</td>
<td>0.393</td>
<td>0.297</td>
<td>0.416</td>
<td>0.317</td>
</tr>
<tr>
<td>LIDNN</td>
<td>ListMLE</td>
<td>0.427</td>
<td>0.219</td>
<td>0.435</td>
<td>0.325</td>
<td>0.455</td>
<td>0.344</td>
</tr>
<tr>
<td></td>
<td>SoftRank</td>
<td>0.457</td>
<td>0.234</td>
<td>0.445</td>
<td>0.336</td>
<td>0.464</td>
<td>0.355</td>
</tr>
<tr>
<td></td>
<td>AttRank</td>
<td>0.455</td>
<td>0.237</td>
<td>0.436</td>
<td>0.334</td>
<td>0.458</td>
<td>0.354</td>
</tr>
<tr>
<td>DLCM</td>
<td>ListMLE</td>
<td>0.457</td>
<td>0.235</td>
<td>0.445</td>
<td>0.336</td>
<td>0.464</td>
<td>0.355</td>
</tr>
<tr>
<td></td>
<td>SoftRank</td>
<td><strong>0.463</strong></td>
<td>0.243</td>
<td>0.447</td>
<td>0.342</td>
<td>0.465</td>
<td>0.360</td>
</tr>
<tr>
<td></td>
<td>AttRank</td>
<td><strong>0.463</strong></td>
<td>0.246</td>
<td><strong>0.450</strong></td>
<td>0.344</td>
<td><strong>0.469</strong></td>
<td><strong>0.362</strong></td>
</tr>
<tr>
<td></td>
<td>LambdaMART</td>
<td>0.457</td>
<td>0.235</td>
<td>0.445</td>
<td>0.336</td>
<td>0.464</td>
<td>0.355</td>
</tr>
</tbody>
</table>
# Ranking Performance on LambdaMART

<table>
<thead>
<tr>
<th>Model</th>
<th>Loss Func</th>
<th>NDCG@1</th>
<th>ERR@1</th>
<th>NDCG@5</th>
<th>ERR@5</th>
<th>NDCG@10</th>
<th>ERR@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>ListMLE</td>
<td>0.372</td>
<td>0.174</td>
<td>0.386</td>
<td>0.278</td>
<td>0.409</td>
<td>0.299</td>
</tr>
<tr>
<td></td>
<td>SoftRank</td>
<td>0.384</td>
<td>0.209</td>
<td>0.378</td>
<td>0.302</td>
<td>0.397</td>
<td>0.321</td>
</tr>
<tr>
<td></td>
<td>AttRank</td>
<td>0.388</td>
<td>0.199</td>
<td>0.393</td>
<td>0.297</td>
<td>0.416</td>
<td>0.317</td>
</tr>
<tr>
<td>LIDNN</td>
<td>ListMLE</td>
<td>0.427</td>
<td>0.219</td>
<td>0.435</td>
<td>0.325</td>
<td>0.455</td>
<td>0.344</td>
</tr>
<tr>
<td></td>
<td>SoftRank</td>
<td>0.457</td>
<td>0.234</td>
<td>0.445</td>
<td>0.336</td>
<td>0.464</td>
<td>0.355</td>
</tr>
<tr>
<td></td>
<td>AttRank</td>
<td>0.455</td>
<td>0.237</td>
<td>0.436</td>
<td>0.334</td>
<td>0.458</td>
<td>0.354</td>
</tr>
<tr>
<td>DLCM</td>
<td>ListMLE</td>
<td>0.457</td>
<td>0.235</td>
<td>0.445</td>
<td>0.336</td>
<td>0.464</td>
<td>0.355</td>
</tr>
<tr>
<td></td>
<td>SoftRank</td>
<td>0.463</td>
<td>0.243</td>
<td>0.447</td>
<td>0.342</td>
<td>0.465</td>
<td>0.360</td>
</tr>
<tr>
<td></td>
<td>AttRank</td>
<td>0.463</td>
<td>0.246</td>
<td>0.450</td>
<td>0.344</td>
<td>0.469</td>
<td>0.362</td>
</tr>
<tr>
<td></td>
<td>LambdaMART</td>
<td>0.457</td>
<td>0.235</td>
<td>0.445</td>
<td>0.336</td>
<td>0.464</td>
<td>0.355</td>
</tr>
</tbody>
</table>

* Microsoft 30K

- Observation
  - DNN damaged the overall ranking performance.
  - DLCM still significantly improved the initial ranker.
# Ranking Performance on LambdaMART

* Microsoft 30K

<table>
<thead>
<tr>
<th>Model</th>
<th>Loss Func</th>
<th>NDCG@1</th>
<th>ERR@1</th>
<th>NDCG@5</th>
<th>ERR@5</th>
<th>NDCG@10</th>
<th>ERR@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>ListMLE</td>
<td>0.372</td>
<td>0.174</td>
<td>0.386</td>
<td>0.278</td>
<td>0.409</td>
<td>0.299</td>
</tr>
<tr>
<td></td>
<td>SoftRank</td>
<td>0.384</td>
<td>0.209</td>
<td>0.378</td>
<td>0.302</td>
<td>0.397</td>
<td>0.321</td>
</tr>
<tr>
<td></td>
<td>AttRank</td>
<td>0.388</td>
<td>0.199</td>
<td>0.393</td>
<td>0.297</td>
<td>0.416</td>
<td>0.317</td>
</tr>
<tr>
<td>LIDNN</td>
<td>ListMLE</td>
<td>0.427</td>
<td>0.219</td>
<td>0.435</td>
<td>0.325</td>
<td>0.455</td>
<td>0.344</td>
</tr>
<tr>
<td></td>
<td>SoftRank</td>
<td>0.457</td>
<td>0.234</td>
<td>0.445</td>
<td>0.336</td>
<td>0.464</td>
<td>0.355</td>
</tr>
<tr>
<td></td>
<td>AttRank</td>
<td>0.455</td>
<td>0.237</td>
<td>0.436</td>
<td>0.334</td>
<td>0.458</td>
<td>0.354</td>
</tr>
<tr>
<td>DLCM</td>
<td>ListMLE</td>
<td>0.457</td>
<td>0.235</td>
<td>0.445</td>
<td>0.336</td>
<td>0.464</td>
<td>0.355</td>
</tr>
<tr>
<td></td>
<td>SoftRank</td>
<td>0.463</td>
<td>0.243</td>
<td>0.447</td>
<td>0.342</td>
<td>0.465</td>
<td>0.360</td>
</tr>
<tr>
<td></td>
<td>AttRank</td>
<td>0.463</td>
<td>0.246</td>
<td>0.450</td>
<td>0.344</td>
<td>0.469</td>
<td>0.362</td>
</tr>
<tr>
<td>LambdaMART</td>
<td></td>
<td>0.457</td>
<td>0.235</td>
<td>0.445</td>
<td>0.336</td>
<td>0.464</td>
<td>0.355</td>
</tr>
</tbody>
</table>

- **Observation**
  - SoftRank and AttRank are better than ListMLE.
  - AttRank is 2 and 20 times faster than ListMLE and SoftRank.
Summary

• We formulate the problem of *Learning to Rank with Local Context*.

• We develop a *Deep Listwise Context Model* that can refine the results of a LTR system with local context.
  – Our method work for all types of LTR features, and require no additional feature extraction.
  – We propose an *attention-based listwise loss function* that is effective and efficient.

• Our experiments demonstrate that incorporating local context is beneficial for learning to rank.
Thanks!

Q&A

aiqy@cs.umass.edu
http://www.cs.umass.edu/~aiqy/
Parameter Sensitivity

![Graphs showing parameter sensitivity](image)
Reference


