Stochastic Top-k ListNet

Tianyi Luo, Dong Wang, Rong Liu, Yiqiao Pan
CSLT / RIIT
Tsinghua University
lty@cslt.riit.tsinghua.edu.cn

EMNLP, Sep 9-19, 2015
Score function \( d^{(i)}: i\)-th document; \( q \): query):

\[
f(d^{(i)}, q) = \omega_1 \cdot \text{feature1}(d^{(i)}) + \omega_2 \cdot \text{feature2}(d^{(i)})
\]

Documents: \( d^{(1)} \) (relevant); \( d^{(2)} \) (irrelevant).

And we set \( \omega_1 = 1, \omega_2 = 1 \).

\[
f(d^{(1)}) = 1 \times 3 + 1 \times 4 = 7; \quad f(d^{(2)}) = 1 \times 5 + 1 \times 3 = 8.
\]

(relevant) \( f(d^{(1)}) = 7 < 8 = f(d^{(2)}) \) (irrelevant) No good rank!
Sample of Learning to Rank

• Score function($d^{(i)}$: $i$-th document; $q$ : query):

\[ f(d^{(i)}, q) = \omega_1 \cdot feature1(d^{(i)}) + \omega_2 \cdot feature2(d^{(i)}) \]

• Documents: $d^{(1)}$ (relevant); $d^{(2)}$ (irrelevant).

Through learning to rank, we get new weights $\omega_1 = 0.1, \omega_2 = 0.9$.

• $f(d^{(1)}) = 0.1 \times 3 + 0.9 \times 4 = 3.9$; $f(d^{(2)}) = 0.1 \times 5 + 0.9 \times 3 = 3.2$.

• (relevant) $f(d^{(1)}) = 3.9 > 3.2 = f(d^{(2)})$ (irrelevant) Good rank!
Wide Applications of Learning to Rank

- Document retrieval
- Question answering
- Text summarization
- Machine translation
- Online advertising
- Collaborative filtering
• Categorization of the Learning to Rank algorithms
  ✶ Pointwise Approach (e.g. McRank, Pranking)
  ✶ Pairwise Approach (e.g. Ranking SVM, RankBoost, Frank)
  ✶ Listwise Approach (e.g. ListNet, ListMLE, AdaRank, SVM\textsuperscript{map})
Listwise Learning to Rank

• Categorization of the Learning to Rank algorithms
  ✧ Pointwise Approach (e.g. McRank, Pranking)
  ✧ Pairwise Approach (e.g. Ranking SVM, RankBoost, Frank)
  ✧ Listwise Approach (e.g. ListNet, ListMLE, AdaRank, SVM$^\text{map}$)

• **Listwise learning** delivers **better** performance in general than **pointwise and pairwise** learning.

(Liu, 2009)
Winning numbers of 7 different learning to rank algorithms

\[ S_i(M) = \sum_{j=1}^{7} \sum_{k=1}^{7} I\{M_i(j) > M_k(j)\} \]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>N@1</th>
<th>N@3</th>
<th>N@10</th>
<th>P@1</th>
<th>P@3</th>
<th>P@10</th>
<th>MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>RankSVM</td>
<td>21</td>
<td>22</td>
<td>22</td>
<td>21</td>
<td>22</td>
<td>22</td>
<td>24</td>
</tr>
<tr>
<td>RankBoost</td>
<td>18</td>
<td>22</td>
<td>22</td>
<td>17</td>
<td>22</td>
<td>23</td>
<td>19</td>
</tr>
<tr>
<td>FRank</td>
<td>18</td>
<td>19</td>
<td>18</td>
<td>18</td>
<td>17</td>
<td>23</td>
<td>15</td>
</tr>
<tr>
<td>ListNet</td>
<td>29</td>
<td>31</td>
<td>33</td>
<td>30</td>
<td>32</td>
<td>35</td>
<td>33</td>
</tr>
<tr>
<td>AdaRank</td>
<td>26</td>
<td>25</td>
<td>26</td>
<td>23</td>
<td>22</td>
<td>16</td>
<td>27</td>
</tr>
<tr>
<td>SVM\text{map}</td>
<td>23</td>
<td>24</td>
<td>22</td>
<td>25</td>
<td>20</td>
<td>17</td>
<td>25</td>
</tr>
</tbody>
</table>
Winning numbers of 7 different learning to rank algorithms

\[ S_i(M) = \sum_{j=1}^{7} \sum_{k=1}^{7} I\{M_i(j) > M_k(j)\} \]

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>N@1</th>
<th>N@3</th>
<th>N@10</th>
<th>P@1</th>
<th>P@3</th>
<th>P@10</th>
<th>MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>RankSVM</td>
<td>21</td>
<td>22</td>
<td>22</td>
<td>21</td>
<td>22</td>
<td>22</td>
<td>24</td>
</tr>
<tr>
<td>RankBoost</td>
<td>18</td>
<td>22</td>
<td>22</td>
<td>17</td>
<td>22</td>
<td>23</td>
<td>19</td>
</tr>
<tr>
<td>FRank</td>
<td>18</td>
<td>19</td>
<td>18</td>
<td>18</td>
<td>17</td>
<td>23</td>
<td>15</td>
</tr>
<tr>
<td><strong>ListNet</strong></td>
<td><strong>29</strong></td>
<td><strong>31</strong></td>
<td><strong>33</strong></td>
<td><strong>30</strong></td>
<td><strong>32</strong></td>
<td><strong>35</strong></td>
<td><strong>33</strong></td>
</tr>
<tr>
<td>AdaRank</td>
<td>26</td>
<td>25</td>
<td>26</td>
<td>23</td>
<td>22</td>
<td>16</td>
<td>27</td>
</tr>
<tr>
<td>SVM^{map}</td>
<td>23</td>
<td>24</td>
<td>22</td>
<td>25</td>
<td>20</td>
<td>17</td>
<td>25</td>
</tr>
</tbody>
</table>
Limitations of Top-k ListNet

• Conventional ListNet: too complex to use in practice
Limitations of Top-k ListNet

• Conventional ListNet: too complex to use in practice

• Top-k ListNet:
  ♦ Top-k ListNet is a optimal algorithm of conventional ListNet
  ♦ However, when k is large, the complexity is still high.
  ♦ When k is small, a lot of ranking information is lost.
Limitations of Top-k ListNet

• Conventional ListNet: too complex to use in practice

• Top-k ListNet:
  ✧ Top-k ListNet is an optimal algorithm of conventional ListNet.
  ✧ However, when k is large, the complexity is still high.
  ✧ When k is small, a lot of ranking information is lost.

Goal: Reduce the computational complexity of Top-k ListNet
• Top-1 ListNet implemented by Cao et al. (2007)
Related work and our work

• Top-1 ListNet implemented by Cao et al. (2007)

• Top-1 ListNet implemented by Dang (2013)
• Top-1 ListNet implemented by Cao et al. (2007)
• Top-1 ListNet implemented by Dang (2013)
• Pairwise l2r sampling implemented by Sculley et al. (2009)
Related work and our work

- Top-1 ListNet implemented by Cao et al. (2007)
- Top-1 ListNet implemented by Dang (2013)
- Pairwise l2r sampling implemented by Sculley et al. (2009)
- Our work:

  Apply sampling methods in Top-k ListNet and publish the code
• Challenges and Motivation
• Model
• Experiments
• Analysis
• Conclusion
ListNet (Cao, 2007)

- Non-measure specific listwise ranking
- Ranked list $\leftrightarrow$ Permutation probability distribution
- Permutation and ranked list: 1-1 correspondence
ListNet (Cao, 2007)

- Non-measure specific listwise ranking
- Ranked list $\leftrightarrow$ Permutation probability distribution
- Permutation and ranked list: 1-1 correspondence

\[ f: f(A)=3, f(B)=0, f(C)=1; \]

Probability distribution of permutations of ABC
ListNet (Cao, 2007)

• Non-measure specific listwise ranking
• Ranked list $\leftrightarrow$ Permutation probability distribution
• Permutation and ranked list: 1-1 correspondence

$f$: $f(A)=3, f(B)=0, f(C)=1$

Probability distribution $\leftrightarrow$ of permutations of ABC
• Probability of a permutation is defined with Luce model:

\[ P(\pi|f) = \prod_{j=1}^{m} \frac{\varphi(f(x_{\pi(j)}))}{\sum_{k=j}^{m} \varphi(f(x_{\pi(k)}))} \]

• Example:

\[ P(ABC|f) = \frac{\varphi(f(A))}{\varphi(f(A))+\varphi(f(B))+\varphi(f(C))} \cdot \frac{\varphi(f(B))}{\varphi(f(B))+\varphi(f(C))} \cdot \frac{\varphi(f(C))}{\varphi(f(C))} \]
Probability of a permutation is defined with Luce model:

\[ P(\pi|f) = \prod_{j=1}^{m} \frac{\varphi(f(x_{\pi(j)}))}{\sum_{k=j}^{m} \varphi(f(x_{\pi(k)}))} \]

Example:

\[ P(ABC|f) = \frac{\varphi(f(A))}{\varphi(f(A)) + \varphi(f(B)) + \varphi(f(C))} \cdot \frac{\varphi(f(B))}{\varphi(f(B)) + \varphi(f(C))} \cdot \frac{\varphi(f(C))}{\varphi(f(C))} \]

P(A ranked No.1)
• Probability of a permutation is defined with Luce model:

\[ P(\pi | f) = \prod_{j=1}^{m} \frac{\phi(f(x_{\pi(j)}))}{\sum_{k=j}^{m} \phi(f(x_{\pi(k)}))} \]

• Example:

\[ P(ABC | f) = \frac{\phi(f(A))}{\phi(f(A)) + \phi(f(B)) + \phi(f(C))} \cdot \frac{\phi(f(B))}{\phi(f(B)) + \phi(f(C))} \cdot \frac{\phi(f(C))}{\phi(f(C))} \]

\[ P(A \text{ ranked No.1}) \]

\[ P(B \text{ ranked No.2 | A ranked No.1}) \]
• Probability of a permutation is defined with Luce model:

\[ P(\pi|f) = \prod_{j=1}^{m} \frac{\phi(f(x_{\pi(j)}))}{\sum_{k=j}^{m} \phi(f(x_{\pi(k)}))} \]

• Example:

\[ P(ABC|f) = \frac{\phi(f(A))}{\phi(f(A)) + \phi(f(B)) + \phi(f(C))} \cdot \frac{\phi(f(B))}{\phi(f(B)) + \phi(f(C))} \cdot \frac{\phi(f(C))}{\phi(f(C))} \]

\[ P(A \text{ ranked No.1}) \]
\[ P(B \text{ ranked No.2} | A \text{ ranked No.1}) \]
\[ P(C \text{ ranked No.3} | A \text{ ranked No.1}, B \text{ ranked No.2}) \]
f: f(A)=3, f(B)=0, f(C)=1;  
Probability distribution ↔ of permutations of ABC
f: f(A)=3, f(B)=0, f(C)=1;
Probability distribution ↔
of permutations of ABC

- Luce model is continuous, differentiable, and concave.
- As shown above, $P(ACB)$ is largest and $P(BCA)$ is smallest.
- Translation invariant, e.g. $\varphi(s) = exp(s)$
- Scale invariant, e.g. $\varphi(s) = a \cdot s$
ListNet (Cao, 2007)

- Loss function based on KL-divergence between two permutation probability distributions ($\varphi = \exp$)

\[
L = - \sum_i \sum_{g \in G} \left\{ P_{y^{(i)}}(g) \log \left( P_{z^{(i)}}(g) \right) \right\}
\]

- Model = Neural Network
- Algorithm = Stochastic Gradient Descent

Motivation | Model | Experiments | Discussion | Conclusion
• Loss function of conventional ListNet consider all the permutations:

\[
L = - \sum_i \sum_{g \in G} \left\{ P_{y^{(i)}(f_\omega)}(g) \log \left( P_{z^{(i)}(f_\omega)}(g) \right) \right\}
\]
Top-k ListNet (Cao, 2007)

• Loss function of conventional ListNet consider all the permutations:

\[
L = - \sum_{i} \sum_{g \in G} \left\{ P_{y^{(i)}(f_\omega)}(g) \log \left( P_{z^{(i)}(f_\omega)}(g) \right) \right\}
\]

\[
L = - \sum_{i} \sum_{g \in G_k} \left\{ P_{y^{(i)}(f_\omega)}(g) \log \left( P_{z^{(i)}(f_\omega)}(g) \right) \right\}
\]
• Loss function of conventional ListNet consider all the permutations:

$$L = - \sum_i \sum_{g \in G} \left\{ P_{y^{(i)}(f_\omega)}(g) \log \left( P_{z^{(i)}(f_\omega)}(g) \right) \right\}$$

$$L = - \sum_i \sum_{g \in G_k} \left\{ P_{y^{(i)}(f_\omega)}(g) \log \left( P_{z^{(i)}(f_\omega)}(g) \right) \right\}$$

where $G_k(j_1, j_2, ... j_k) = \{ \pi \in \Omega_n | \pi(t) = j_t, \forall t = 1, 2, ..., k \}$

Conventional ListNet = Top–k ListNet when $k = n$
• Loss function of conventional ListNet consider all the permutations:

\[
L = - \sum_i \sum_{g \in G} \left\{ P_{y(i)}(f_\omega)(g) \log \left( P_{z(i)}(f_\omega)(g) \right) \right\}
\]

\[
L = - \sum_i \sum_{g \in G_k} \left\{ P_{y(i)}(f_\omega)(g) \log \left( P_{z(i)}(f_\omega)(g) \right) \right\}
\]

\[
L = - \sum_i \sum_{g \in G_k} \left\{ P_{y(i)}(f_\omega)(g) \log \left( P_{z(i)}(f_\omega)(g) \right) \right\}
\]

where \( G_k (j_1, j_2, ... j_k) = \{ \pi \in \Omega_n | \pi(t) = j_t, \forall t = 1, 2, ..., k \} \)
Limitation of Top-k ListNet (Cao, 2007)

• Challenge 1: Computational complexity
• Challenge 1: Computational complexity
  ✧ Reduce the complexity for query m: $n_m! \rightarrow \frac{n_m!}{(n_m-k)!}$

  ✧ However, when k is large, the complexity is still high.

  ✧ When k is small, a lot of ranking information is lost.

  ✧ When existing queries which have lots of candidates, k=2 is large.
Challenge 1: Computational complexity

▷ Reduce the complexity for query m: \( n_m! \rightarrow \frac{n_m!}{(n_m-k)!} \)

▷ However, when k is large, the complexity is still high.

▷ When k is small, a lot of ranking information is lost.

▷ When existing queries which have lots of candidates, k=2 is large.

Challenge 2: Data Imbalance

Limitation of Top-k ListNet (Cao, 2007)
• Challenge 1: Computational complexity

✧ Reduce the complexity for query m: $n_m! → \frac{n_m!}{(n_m-k)!}$

✧ However, when k is large, the complexity is still high.

✧ When k is small, a lot of ranking information is lost.

✧ When existing queries which have lots of candidates, k=2 is large.

• Challenge 2: Data Imbalance

✧ Many irrelevant documents exist in benchmarks e.g. LETOR.

✧ More list containing relevant documents should be considered.
• Challenges:

- Computational complexity
- Data Imbalance
• Challenges:

- Computational complexity
- Data Imbalance

Our solution: Sampling and Resampling!
• To overcome Challenge 1: Computational complexity
• Sample a small set of the Top-k permutation lists.
• Impose a bound of the computation cost.
• To overcome Challenge 1: Computational complexity
• Sample a small set of the Top-k permutation lists.
• Impose a bound of the computation cost.
• Three sampling methods (e.g. $k = 3$):

  ✷ Uniform distribution sampling

  ✷ Fixed distribution sampling

  ✷ Adaptive distribution sampling
• Three sampling methods (e.g. $k = 3$):

- Uniform distribution sampling

1. Sampling $l$ lists (Every list is selected with the same probability;
2. Probability distribution doesn’t change every time.)
Three sampling methods (e.g. \( k = 3 \)):

- **Uniform distribution sampling:**
  
  1. Sampling \( l \) lists (Every list is selected with the **same** probability;)
  2. Probability **distribution doesn’t change every time.**

<table>
<thead>
<tr>
<th>Probability Distribution:</th>
<th>Probability Distribution:</th>
<th>Probability Distribution:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC:1/6  ACB:1/6</td>
<td>ABC:1/6  ACB:1/6</td>
<td>ABC:1/6  ACB:1/6</td>
</tr>
<tr>
<td>BAC:1/6  BCA:1/6</td>
<td>BAC:1/6  BCA:1/6</td>
<td>BAC:1/6  BCA:1/6</td>
</tr>
<tr>
<td>CAB:1/6  CBA:1/6</td>
<td>CAB:1/6  CBA:1/6</td>
<td>CAB:1/6  CBA:1/6</td>
</tr>
</tbody>
</table>

One sample list: ABC  
Permutations will be selected randomly.
Three sampling methods (e.g. k = 3):

- Fixed distribution sampling

1. Sampling l lists (Every list is selected with the different probabilities;)
2. Different probabilities are based on different labeled relevant degrees;
3. Probability distribution doesn’t change every time.
• Three sampling methods (e.g. $k = 3$):

- Fixed distribution sampling:
  1. Sampling $l$ lists (Every list is selected with the different probabilities);
  2. Different probabilities are based on different labeled relevant degrees;
  3. Probability distribution doesn’t change every time.

<table>
<thead>
<tr>
<th>Probability Distribution:</th>
<th>Probability Distribution:</th>
<th>Probability Distribution:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC:1/4</td>
<td>ACB:1/3</td>
<td>ABC:1/4</td>
</tr>
<tr>
<td>CAB:1/6</td>
<td>CBA:1/8</td>
<td>CAB:1/6</td>
</tr>
</tbody>
</table>

One sample list: ACB

One sample list: ACB

One sample list: ACB

Permutation which has larger probability will be selected easily.
• Three sampling methods (e.g. k = 3):

◊ Adaptive distribution sampling

1. Sampling lists (Every list is selected with the different probabilities);
2. Different probabilities are based on different output scores;
3. Probability distribution adaptively changes every time.
• Three sampling methods (e.g. \(k = 3\)):

✧ Adaptive distribution sampling:

1. Sampling \(l\) lists (Every list is selected with the **different** probabilities;)
2. Different **probabilities** are **based on** different **output scores**;
3. Probability **distribution adaptively changes** every time.

<table>
<thead>
<tr>
<th>List</th>
<th>Probability Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC:1/4</td>
<td>ACB:1/3</td>
</tr>
<tr>
<td>BAC:1/9</td>
<td>BCA:1/24</td>
</tr>
<tr>
<td>CAB:1/6</td>
<td>CBA:1/8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List</th>
<th>Probability Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC:10/36</td>
<td>ACB:9/24</td>
</tr>
<tr>
<td>BAC:1/12</td>
<td>BCA:1/36</td>
</tr>
<tr>
<td>CAB:13/72</td>
<td>CBA:1/12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List</th>
<th>Probability Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC:7/24</td>
<td>ACB:7/18</td>
</tr>
<tr>
<td>BAC:7/72</td>
<td>BCA:1/72</td>
</tr>
<tr>
<td>CAB:7/36</td>
<td>CBA:5/72</td>
</tr>
</tbody>
</table>

One sample list: ACB

Permutation which has **larger** probability will be selected easily.
• To overcome Challenge 2: Data Imbalance

• Re-sampling methods(e.g. k = 3):

\[
\sum_{i=1}^{k} s_i \over kS
\]
Re-sampling

• To overcome Challenge 2: Data Imbalance

• Re-sampling methods (e.g. $k = 3$):

$$\frac{\sum_{i=1}^{k} s_i}{kS}$$

$s_i$: The value of the human-labelled score of the $i$ document selected.

$S$: The maximum value of the human-labelled score, which is 2 in our case.
• To overcome Challenge 2: Data Imbalance

• Re-sampling methods (e.g. k = 3):

\[ \frac{\sum_{i=1}^{k} s_i}{kS} \]

- \( s_i \): The value of the human-labelled score of the \( i \) document selected.
- \( S \): The maximum value of the human-labelled score, which is 2 in our case.

The probability that a list is retained.
To overcome Challenge 2: Data Imbalance

Re-sampling methods (e.g. $k = 3$):

\[
\sum_{i=1}^{k} s_i \times \frac{1}{kS}
\]

$s_i$: The value of the human-labelled score of the $i$ document selected.

$S$: The maximum value of the human-labelled score, which is 2 in our case.

The probability that a list is retained.

Give one example (A: 2 very relevant; B: 1 relevant; C: 0 irrelevant):

Given $s_1 = 2, s_2 = 1, s_3 = 0, \sum_{i=1}^{k} s_i = \frac{2+1+0}{3+2} = \frac{1}{2}$
• Re-sampling steps:
  ✷ Calculate the probability of ABC. It is 1/2 based on last slide.
  ✷ Randomly generate a value in (0,1) e.g. 1/6.
  ✷ If 1/6 < 1/2, list ABC is retained to conduct training.

• Re-sampling make lists containing more relevant documents trained
• When k=1:

\[ \Delta \omega = \sum_j \left\{ \sigma(z^{(i),j}) - \sigma(y^{(i),j}) \right\} \cdot x^{(i)}_j \]

Where \( \sigma(s,j) \) is the j-th value of the softmax function of the score vector \( s \), given by:

\[ \sigma(s^{(i),j}) = \frac{e^{s^{j,i}}}{\sum_{t=1}^{n^{(i)}} e^{s_{t}^{(i)}}} \]
• When \( k > 1 \):

\[
\Delta \omega = \sum_{g \in G_k} \left( \sum_{t=1}^{k} \hat{\sigma}(y^{(i)}, t) \right) \cdot \left[ \sum_{f=1}^{k} \left( x_{jf} - \sum_{v=C}^{v} \hat{\sigma}(z^{(i)}, v) \cdot x_{jf} \right) \right]
\]

Where \( \hat{\sigma}(\cdot) \) defines a ‘partial’ softmax function of the score vector \( s \), given by:

\[
\hat{\sigma}(s^{(i)}, f) = \frac{e^{s_f^i}}{\sum_{t=C}^{n} e^{s_t^i}}
\]
• Dataset
  ✷ LETOR 4.0 (Liu et al., 2007)
  ✷ We use MQ2008 dataset in the LETOR 4.0
  ✷ Training set, validation set and test data all contain 784 queries.

• Learning rate:
  ✷ 10–3 for $k = 1$; 10–5 for $k > 1$. 
Experimental Results

- $k = 1$:

Figure 1: The P@1 performance on the test data with the Top-1 ListNet utilizing the three sampling approaches. The size of the permutation subset varies from 50 to 500.
Experimental Results

- $k = 2$:

Figure 2: The P@1 performance on the test data with the Top-2 ListNet utilizing the three sampling approaches. The size of the permutation subset varies from 5 to 500.
• $k = 3$:

Figure 3: The P@1 performance on the test data with the Top-3 ListNet utilizing the three sampling approaches. The size of the permutation subset varies from 5 to 500.
• $k = 4$:

Figure 4: The P@1 performance on the test data with the Top-4 ListNet utilizing the three sampling approaches. The size of the permutation subset varies from 5 to 500.
• Larger k:

Figure 5: The P@1 performance on the test data with the stochastic Top-k ListNet approach, where $k$ varies from 1 to 100.
Experimental Results

• P@1 and P@10:

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-k</th>
<th>Sampling</th>
<th>Time (s)</th>
<th>P@1</th>
<th>P@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-ListNet</td>
<td>k=1</td>
<td>-</td>
<td>2.509</td>
<td>0.4101</td>
<td>0.4107</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=1</td>
<td>UDS</td>
<td>0.753</td>
<td>0.4097</td>
<td>0.4106</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=1</td>
<td>FDS</td>
<td>0.391</td>
<td>0.4094</td>
<td>0.4090</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=1</td>
<td>ADS</td>
<td><strong>0.375</strong></td>
<td><strong>0.4102</strong></td>
<td></td>
</tr>
<tr>
<td>C-ListNet</td>
<td>k=2</td>
<td>-</td>
<td>2275.5</td>
<td>0.4119</td>
<td>0.4043</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=2</td>
<td>UDS</td>
<td>2.898</td>
<td>0.4140</td>
<td>0.4130</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=2</td>
<td>FDS</td>
<td>2.410</td>
<td>0.4145</td>
<td>0.4144</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=2</td>
<td>ADS</td>
<td><strong>2.013</strong></td>
<td><strong>0.4162</strong></td>
<td></td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=3</td>
<td>UDS</td>
<td>4.358</td>
<td>0.4167</td>
<td>0.4204</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=3</td>
<td>FDS</td>
<td>3.997</td>
<td>0.4137</td>
<td>0.4205</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=3</td>
<td>ADS</td>
<td><strong>3.483</strong></td>
<td><strong>0.4184</strong></td>
<td></td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=4</td>
<td>UDS</td>
<td>6.161</td>
<td>0.4145</td>
<td>0.4226</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=4</td>
<td>FDS</td>
<td>5.773</td>
<td>0.4145</td>
<td>0.4232</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=4</td>
<td>ADS</td>
<td><strong>4.358</strong></td>
<td><strong>0.4149</strong></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Averaged training time (in seconds), P@1 and P@10 on training, validation (Val.) and test data with different Top-k methods. ‘C-ListNet’ stands for conventional ListNet, ‘S-ListNet’ stands for stochastic ListNet.
Experimental Results

• P@1 and P@10:

Table 1: Averaged training time (in seconds), P@1 and P@10 on training, validation (Val.) and test data with different Top-k methods. ‘C-ListNet’ stands for conventional ListNet, ‘S-ListNet’ stands for stochastic ListNet.

<table>
<thead>
<tr>
<th>Model</th>
<th>Top-k</th>
<th>Sampling</th>
<th>Time (s)</th>
<th>P@1 Train</th>
<th>P@1 Val.</th>
<th>P@1 Test</th>
<th>P@10 Train</th>
<th>P@10 Val.</th>
<th>P@10 Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-ListNet</td>
<td>k=1</td>
<td>-</td>
<td>2.509</td>
<td>0.4101</td>
<td>0.4107</td>
<td>0.4119</td>
<td>0.2684</td>
<td>0.2684</td>
<td>0.2676</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=1</td>
<td>UDS</td>
<td>0.753</td>
<td>0.4097</td>
<td>0.4106</td>
<td>0.4120</td>
<td>0.2680</td>
<td>0.2683</td>
<td>0.2676</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=1</td>
<td>FDS</td>
<td>0.391</td>
<td>0.4094</td>
<td>0.4090</td>
<td>0.4127</td>
<td>0.2679</td>
<td>0.2681</td>
<td>0.2676</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=1</td>
<td>ADS</td>
<td>0.375</td>
<td>0.4102</td>
<td>0.4097</td>
<td>0.4121</td>
<td>0.2680</td>
<td>0.2682</td>
<td>0.2677</td>
</tr>
<tr>
<td>C-ListNet</td>
<td>k=2</td>
<td>-</td>
<td>2275.5</td>
<td>0.4119</td>
<td>0.4043</td>
<td>0.4043</td>
<td>0.2678</td>
<td>0.2674</td>
<td>0.2674</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=2</td>
<td>UDS</td>
<td>2.898</td>
<td>0.4140</td>
<td>0.4143</td>
<td>0.4130</td>
<td>0.2682</td>
<td>0.2686</td>
<td>0.2681</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=2</td>
<td>FDS</td>
<td>2.410</td>
<td>0.4145</td>
<td>0.4144</td>
<td>0.4164</td>
<td>0.2684</td>
<td>0.2688</td>
<td>0.2684</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=2</td>
<td>ADS</td>
<td>2.013</td>
<td>0.4162</td>
<td>0.4168</td>
<td>0.4145</td>
<td>0.2686</td>
<td>0.2689</td>
<td>0.2687</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=3</td>
<td>UDS</td>
<td>4.358</td>
<td>0.4167</td>
<td>0.4204</td>
<td>0.4152</td>
<td>0.2686</td>
<td>0.2681</td>
<td>0.2680</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=3</td>
<td>FDS</td>
<td>3.997</td>
<td>0.4137</td>
<td>0.4205</td>
<td>0.4131</td>
<td>0.2687</td>
<td>0.2695</td>
<td>0.2685</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=3</td>
<td>ADS</td>
<td>3.483</td>
<td>0.4184</td>
<td>0.4196</td>
<td>0.4177</td>
<td>0.2692</td>
<td>0.2697</td>
<td>0.2689</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=4</td>
<td>UDS</td>
<td>6.161</td>
<td>0.4145</td>
<td>0.4226</td>
<td>0.4104</td>
<td>0.2686</td>
<td>0.2694</td>
<td>0.2687</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=4</td>
<td>FDS</td>
<td>5.773</td>
<td>0.4145</td>
<td>0.4232</td>
<td>0.4150</td>
<td>0.2690</td>
<td>0.2695</td>
<td>0.2686</td>
</tr>
<tr>
<td>S-ListNet</td>
<td>k=4</td>
<td>ADS</td>
<td>4.358</td>
<td>0.4149</td>
<td>0.4247</td>
<td>0.4164</td>
<td>0.2692</td>
<td>0.2700</td>
<td>0.2689</td>
</tr>
</tbody>
</table>

1000+ times!
• Better measures of document scores
  ✷ Good measures of rank but not good measures of relevance.
  ✷ Better measures can be learnt and make performance better.

• Harsh labelling of the MQ 2008 dataset
  ✷ Documents are labelled by only three values \{0, 1, 2\}.
  ✷ Training data containing many documents with the same scores.

• Other Listwise algorithms can utilize sampling methods
• Summary:

✧ Proposed a stochastic ListNet method.

✧ Speed up the training and improve the rank performance.

• Future work:

✧ Apply our method to other datasets which contains more human labels.

✧ Apply our method to other listwise learning to rank methods.

• Our code:

✧ https://github.com/pkuluotianyi/topKStoListNet
Thank you!

Presented by Tianyi Luo

EMNLP, Sep 9-19, 2015