Ranking Distillation: Learning Compact Ranking Models With High Performance for Recommender System

Jiaxi Tang and Ke Wang
Simon Fraser University
jiaxit@sfu.ca, wangk@cs.sfu.ca
Motivation

- More Powerful
  - Small-size Model
  - Medium-size Model
  - Model with Ranking Distillation

- More Serving Cost
  - Large-size Model
  - Cannot afford!

Too bad result!
Motivation

(a). Larger model size -> Better performance
   *Hard to serve the model for inefficiency!*

(b). More training instances -> Better performance
   *Not always available!*
What is Knowledge Distillation (KD)?

Smaller Model (student)

Learns from dataset

Learns from teacher model

Larger Model (teacher, well-trained)

This is a Cat.

This is most likely a Cat, but it also seems like a Tiger.

Makes the student model more robust, generalizable and thus perform better.
Analogy in Ranking Problem

For this user, the ground-truth ranking is: Item2 > item5 > ...

Smaller Model (student)

Learns to rank from dataset

Larger Model (teacher, well-trained)

I love action movies!

For this user, we should also rank item1, item4 and item9 at top positions!

Action movies

Adventure movies

Dataset

Smaller Model

Learns to rank from teacher
Ranking Distillation

Given Query q

Labeled Document Set $\mathcal{O}$

Unlabeled Document Set $\overline{\mathcal{O}}$

We have very few labels

We do negative sampling

Teacher Model: $M_T$

(well-trained)

Student Model: $M_S$

In recommendation: Query -> User Profile, Document -> Item

Model Predicted Top-K Ranking:

$$\pi_{1..K} = (\pi_1, \pi_2, ..., \pi_K)$$

Compute Ranking Loss: $\mathcal{L}^R$

Compute Distillation Loss: $\mathcal{L}^D$

Forward Propagation

Backward Propagation

Traditional Module

Distillation Module

We have very few labels
Ranking Distillation

Loss for a single query:
\[ \mathcal{L}(\theta_S) = (1 - \alpha) \mathcal{L}^R(y, \hat{y}) + \alpha \mathcal{L}^D(\pi_{1..K}, \hat{y}). \]

\( \mathcal{L}^R(y, \hat{y}) = - \sum_{d \in y_d^+} \log(P(\text{rel} = 1|\hat{y}_d)) + \sum_{d \in y_d^-} \log(1 - P(\text{rel} = 1|\hat{y}_d)) \)

\( \mathcal{L}^D(\pi_{1..K}, \hat{y}) = - \sum_{r=1}^{K} w_r \cdot \log(P(\text{rel} = 1|\hat{y}_{\pi_r})) \)

\( = - \sum_{r=1}^{K} w_r \cdot \log(\sigma(\hat{y}_{\pi_r})) \)

Pros: simple, only consider positive documents.
Cons: non-differentiable.

Weighting by position importance $w^a$

**Assumption**: The teacher predicted unlabeled documents at top positions are more correlated to the query and are more likely to the positive ground-truth documents.

An empirical weight following a exponentially decayed function[1]:

$$w^a_r \propto e^{-r/\lambda} \quad \text{and} \quad \lambda \in \mathbb{R}^+$$

Weighting by ranking discrepancy $w^b$

Assumption: During the training process, we should have a dynamic weight to upweight the erroneous parts in distillation loss, and downweight the parts that already learned perfectly.

<table>
<thead>
<tr>
<th>Teacher’s rank</th>
<th>Student’s rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_1$</td>
<td>1</td>
</tr>
<tr>
<td>$\pi_2$</td>
<td>2</td>
</tr>
<tr>
<td>$\pi_3$</td>
<td>3</td>
</tr>
</tbody>
</table>

Example:

$$L^D = w_1^b \cdot \log(\hat{y}_{\pi_1}) + w_2^b \cdot \log(\hat{y}_{\pi_2}) + w_3^b \cdot \log(\hat{y}_{\pi_3})$$

How do we know student’s rank without computing relevance scores for all items?

To get a documents approximated rank in a list of $N$ documents, we can randomly sample $\epsilon \ll N$ documents in this list, and:

Estimated rank = $\lceil n \times (N - 1)/\epsilon \rceil + 1$

where $n$ is the number of documents whose scores are greater than the given documents score.

$$w_3^b = \tanh(\mu \times (156 - 3))$$
Ranking Distillation by both weights

1. Choose a proper $K$, e.g. $K=3$

2. Using $w^\alpha$ during the first few iterations

3. Using hybrid weights then.

$$w_i = w_i^\alpha \cdot w_i^\beta$$

(normalize (optional))

$w_i \propto w_i^\alpha \cdot w_i^\beta$
Experimental results

• Task: Sequential Recommendation, query -> user & her/his sequence document -> item

• Datasets: Gowalla & Foursquare


• Baselines:
  o Model-T: Teacher model
  o Model-S: Student model
  o Model-RD: Student model trained with ranking distillation

<table>
<thead>
<tr>
<th>Datasets</th>
<th>#users</th>
<th>#items</th>
<th>avg. actions per user</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gowalla</td>
<td>13.1k</td>
<td>14.0k</td>
<td>40.74</td>
</tr>
<tr>
<td>Foursquare</td>
<td>10.1k</td>
<td>23.4k</td>
<td>30.16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Model</th>
<th>Time (CPU)</th>
<th>Time (GPU)</th>
<th>#Params</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gowalla</td>
<td>Fossil-T</td>
<td>9.32s</td>
<td>3.72s</td>
<td>1.48M</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Fossil-RD</td>
<td>4.99s</td>
<td>2.11s</td>
<td>0.64M</td>
<td>43.2%</td>
</tr>
<tr>
<td></td>
<td>Caser-T</td>
<td>38.58s</td>
<td>4.52s</td>
<td>5.58M</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Caser-RD</td>
<td>18.63s</td>
<td>2.99s</td>
<td>2.79M</td>
<td>50.0%</td>
</tr>
<tr>
<td>Foursquare</td>
<td>Fossil-T</td>
<td>6.35s</td>
<td>2.47s</td>
<td>1.01M</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Fossil-RD</td>
<td>3.86s</td>
<td>2.01s</td>
<td>0.54M</td>
<td>53.5%</td>
</tr>
<tr>
<td></td>
<td>Caser-T</td>
<td>23.89s</td>
<td>2.95s</td>
<td>4.06M</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Caser-RD</td>
<td>11.65s</td>
<td>1.96s</td>
<td>1.64M</td>
<td>40.4%</td>
</tr>
</tbody>
</table>

Experimental results

<table>
<thead>
<tr>
<th>Model</th>
<th>Prec@3</th>
<th>Prec@5</th>
<th>Prec@10</th>
<th>nDCG@3</th>
<th>nDCG@5</th>
<th>nDCG@10</th>
<th>MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fossil-T</td>
<td>0.1299</td>
<td>0.1062</td>
<td>0.0791</td>
<td>0.1429</td>
<td>0.1270</td>
<td>0.1140</td>
<td>0.0866</td>
</tr>
<tr>
<td>Fossil-RD</td>
<td>0.1355</td>
<td>0.1096</td>
<td>0.0808</td>
<td>0.1490</td>
<td>0.1314</td>
<td>0.1172</td>
<td>0.0874</td>
</tr>
<tr>
<td>Fossil-S</td>
<td>0.1217</td>
<td>0.0995</td>
<td>0.0739</td>
<td>0.1335</td>
<td>0.1185</td>
<td>0.1060</td>
<td>0.0792</td>
</tr>
<tr>
<td>Caser-T</td>
<td>0.1408</td>
<td>0.1149</td>
<td>0.0856</td>
<td>0.1546</td>
<td>0.1376</td>
<td>0.1251</td>
<td>0.0958</td>
</tr>
<tr>
<td>Caser-RD</td>
<td>0.1458</td>
<td>0.1183</td>
<td>0.0878</td>
<td>0.1603</td>
<td>0.1423</td>
<td>0.1283</td>
<td>0.0969</td>
</tr>
<tr>
<td>Caser-S</td>
<td>0.1333</td>
<td>0.1094</td>
<td>0.0818</td>
<td>0.1456</td>
<td>0.1304</td>
<td>0.1188</td>
<td>0.0919</td>
</tr>
</tbody>
</table>

Summarize: Models trained with RD have similar performance with their teachers.
Tried but failed

1. Using the Top-$K$ documents from teacher model as positive documents, and using the Bottom-$K$ documents as negative documents. Then apply point-wise, pair-wise, list-wise distillation loss.
   • Possible reason: negative documents can be anywhere except Top-$K$, so we don’t need to care too much about them.

2. Using the pair-wise distillation loss within teacher’s Top-$K$ documents, to make the partial order as correct as possible.
   • Possible reason: the gradient contains both up-wards gradient and down-ward gradient, which cause issues in trainability.

   -- It is ‘good’ that teacher’s ranking and student’s ranking at top positions are not perfectly matched. e.g. teacher’s ranking: $d_1 > d_2 > d_3 > ...$, student’s ranking: $d_2 > d_3 > d_1 > ...$
1. We use the Top-$K$ unlabeled documents from teacher model’s ranking as positive documents, and use a smaller student to learn to rank these documents at higher positions.

2. We propose two different weighting schemes to boost the training process.

3. The proposed ‘Ranking Distillation’ can be regarded as:
   - A knowledge transfering method
   - A semi-supervised method
   - A data-augmentation method
Q&A