TF-Ranking
Learning-to-Rank in TensorFlow

Presenter: Michael Bendersky
(on behalf of the TF-Ranking Team)
Google Research

Search Solutions Nov 25th, 2020
TF-Ranking: TensorFlow Ranking

- Deep learning library for learning-to-rank in TensorFlow
- Open source on GitHub under tensorflow/ranking
- Initial release in Dec. 2018
- Actively developed by the TF-Ranking team at Google Research
Industry Adoption

● Launched in products by many companies
  ○ LinkedIn
  ○ Grubhub
  ○ Zhihu
  ○ iQIYI

● Actively being experimented by
  ○ Uber
  ○ Walmart
  ○ Spotify
  ○ Airbnb
  ○ ...
State of the Art on Public Benchmarks

- MS MARCO **Leaderboard** (as of Nov. 21, 2020)
  - No. 1 for Passage Re Ranking
  - No. 5 for Passage Full Ranking

- **TREC-COVID19**
  - No. 1 in *round 4* for 4 out 5 metrics.
  - No. 1 in *round 5* for all 5 metrics.

<table>
<thead>
<tr>
<th></th>
<th>ndcg@20</th>
<th>P@20</th>
<th>rbp_p5</th>
<th>bpref</th>
<th>map</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.8496</td>
<td>0.8760</td>
<td>0.9197</td>
<td>0.6372</td>
<td>0.4718</td>
</tr>
<tr>
<td></td>
<td>0.8490</td>
<td>0.8690</td>
<td>0.9399</td>
<td>0.6378</td>
<td>0.4731</td>
</tr>
<tr>
<td></td>
<td>0.8311</td>
<td>0.8460</td>
<td>0.9361</td>
<td>0.5330</td>
<td>0.3922</td>
</tr>
<tr>
<td></td>
<td>0.8304</td>
<td>0.8380</td>
<td>0.9370</td>
<td>0.5280</td>
<td>0.3875</td>
</tr>
</tbody>
</table>
Learning-to-Rank (LTR)
**Problem Formulation**

**Problem:** Learning a scoring function $f$ to sort a list of examples
- Input: context, list of examples, labels.
- Output: $f$ that produces the optimal ordering of examples

$$\psi = (x, y) \in X^n \times \mathbb{R}^n$$  \textit{Training sample with relevance labels}

$$\mathcal{L}(f) = \frac{1}{|\Psi|} \sum_{(x, y) \in \Psi} \ell(y, f(x))$$  \textit{Choose $f^*$ to minimize empirical loss}
Ranking Metrics

Standard ranking metrics are either **discontinuous** or **flat** everywhere

- Cannot be directly optimized with gradient descent
Pointwise LTR methods

- Documents are considered independently of each other
- Some examples: *ordinal regression*, *classification*, *GBRT*

\[
\begin{align*}
\text{f ( } & \text{A } \text{) } \rightarrow \text{ P(A is Relevant)} \\
\text{f ( } & \text{B } \text{) } \rightarrow \text{ P(B is Relevant)} \\
\text{f ( } & \text{C } \text{) } \rightarrow \text{ P(C is Relevant)}
\end{align*}
\]
Pairwise LTR methods

- Document pairs are considered
- Some examples: RankNet, RankSVM, RankBoost
Listwise LTR methods

- Consider the ordering of the entire list
- Some examples: LambdaMART, ApproxNDCG, List{Net, MLE}
Traditional LTR Setting

● **Handcrafted** features based on <query, document>
  ○ 136 features in Web30K
    ■ tf-idf scores, BM25 scores
    ■ Inlink counts
    ■ URL length
    ■ Page quality
    ■ ....

● **Human** relevance judgments
  ○ The largest datasets have tens of thousands of labeled examples
    ■ Web30K, Istella, Yahoo! ~30K queries
Why Deep Learning-to-Rank?

- **Sparse features**
  - Directly use query and document keywords as features

- **Large-scale data**
  - User interactions as labels (e.g., clicks)

- **Advance of deep learning technologies**
  - Attention models like Transformer
  - BERT
  - ResNet
  - ...
Challenges Tackled by TF-Ranking

● **Data representation:** How to represent a document list of varying size
  ○ tf.Example is not suitable for a list
  ○ tf.Tensor is not friendly for varying size

● **Losses & Metrics**
  ○ No built-in ranking losses/metrics in TensorFlow
  ○ Implementation should be based on Tensors and tf Ops

● **Serving may differ from Training**
  ○ Training needs the whole list of documents
  ○ Serving only needs a single document (and the query)
Each q, d is a tf.Example

ELWC has 2 fields:
- “context”: q → [a single tf.Example]
- “examples”: [d_1, d_2, …] → [a list of tf.Examples]
Supported Components

- Losses: pointwise/pairwise/listwise losses
- Metrics: MRR, NDCG, MAP, etc.
- Sparse/Embedding features
- Unbiased learning-to-rank from biased data (e.g., clicks)
Supported Loss Examples (Binary Labels)

(Pointwise) Sigmoid Cross Entropy

\[
\hat{\ell}(y, \hat{y}) = -\sum_{j=1}^{n} y_j \log(p_j) + (1 - y_j) \log(1 - p_j)
\]

(Pairwise) Logistic Loss

\[
\hat{\ell}(y, \hat{y}) = \sum_{j=1}^{n} \sum_{k=1}^{n} \mathbb{I}(y_j > y_k) \log(1 + \exp(\hat{y}_k - \hat{y}_j))
\]

(Listwise) Softmax Loss (aka ListNET)

\[
\hat{\ell}(y, \hat{y}) = -\sum_{j=1}^{n} y_j \log\left(\frac{\exp(\hat{y}_j)}{\sum_{j=1}^{n} \exp(\hat{y}_j)}\right)
\]
TF-Ranking - How it works

ELWC

query

doc1 0

doc2 1

doc3 1

label

scoring function

s1
s2
s3

losses

Pointwise:
Minimize: |0 - s1| + |1 - s2| + ...

Pairwise:
Maximize: p(s2>s1) p(s3 > s1) ...

Listwise:
Maximize: ndcg {s1, s2, s3}

Update parameters
New developments in TF-Ranking
New developments

1. **TFR-BERT**
   - Advanced scoring functions

2. **Neural GAMs**
   - Building interpretable & explainable models

3. **Document Interaction Networks**
   - Modeling cross-document interactions
[Han et al. arXiv] Learning-to-Rank with BERT in TF-Ranking.
BERT with Ranking Loss

- The model is fine-tuned by “softmax loss”:

\[ \ell_q = \sum_{d \in C} \frac{y_d}{\sum_{d' \in C} y_{d'}} \log \left( \frac{\exp(s_{C_{BERT}}(d))}{\sum_{d' \in C} \exp(s_{C_{BERT}}(d'))} \right) \]

where \( y_d \) is the ground-truth label

- The loss function considers the other documents in the same list
  - Better ranking performances compared to sigmoid cross-entropy loss

<table>
<thead>
<tr>
<th>Loss Function</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sigmoid CE</td>
<td>37.16</td>
</tr>
<tr>
<td>Pairwise log-loss</td>
<td>37.18</td>
</tr>
<tr>
<td>Softmax Loss</td>
<td>37.82</td>
</tr>
<tr>
<td>Best Ensemble model</td>
<td>38.77</td>
</tr>
</tbody>
</table>

Results on MS-Marco passage re-ranking
Interpretable LTR models: Neural GAM

Capabilities

(a) Distilling sub-models as piecewise curves

(b) Measuring the effect of context features

(a) min of term frequency (whole document)  (b) LMIR.JM (body)  (c) sum of stream length normalized term frequency (whole document)
Performance

- Neural GAM performs better than or on par with other baselines
- Neural GAM handles context features well

<table>
<thead>
<tr>
<th>Data set</th>
<th>Method</th>
<th>NDCG₁</th>
<th>NDCG₅</th>
<th>NDCG₁₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>YAHOO</td>
<td>Tree GAM</td>
<td>67.61</td>
<td>69.46</td>
<td>73.89</td>
</tr>
<tr>
<td></td>
<td>Neural GAM</td>
<td>67.63</td>
<td>69.62</td>
<td>73.98</td>
</tr>
<tr>
<td></td>
<td>Tree RankGAM</td>
<td>69.12</td>
<td>71.03</td>
<td>75.04</td>
</tr>
<tr>
<td></td>
<td>Neural RankGAM</td>
<td>69.36</td>
<td>71.32</td>
<td>75.33*</td>
</tr>
<tr>
<td>WEB30K</td>
<td>Tree GAM</td>
<td>29.79</td>
<td>32.79</td>
<td>35.96</td>
</tr>
<tr>
<td></td>
<td>Neural GAM</td>
<td>30.59</td>
<td>33.55</td>
<td>36.54</td>
</tr>
<tr>
<td></td>
<td>Tree RankGAM</td>
<td>41.90</td>
<td>42.04</td>
<td>44.37</td>
</tr>
<tr>
<td></td>
<td>Neural RankGAM</td>
<td>44.31*</td>
<td>43.29*</td>
<td>45.09*</td>
</tr>
<tr>
<td>CWS</td>
<td>Tree GAM</td>
<td>19.74</td>
<td>32.91</td>
<td>36.72</td>
</tr>
<tr>
<td></td>
<td>Neural GAM</td>
<td>20.09</td>
<td>34.01</td>
<td>38.60</td>
</tr>
<tr>
<td></td>
<td>Tree RankGAM</td>
<td>20.16</td>
<td>35.06</td>
<td>39.27</td>
</tr>
<tr>
<td></td>
<td>Neural RankGAM</td>
<td>20.35</td>
<td>34.94</td>
<td>38.93</td>
</tr>
<tr>
<td></td>
<td>Neural RankGAM+</td>
<td>24.43*</td>
<td>39.88*</td>
<td>42.84*</td>
</tr>
</tbody>
</table>
Permutation Equivariant Document Interaction Network for Neural Learning to Rank

Self Attention Layer

Document Interaction Embedding

Univariate Scoring Function

Document Set

[Pasumarthi et al. ICTIR2020]
## Experiments on Web30K Benchmark

<table>
<thead>
<tr>
<th>Method</th>
<th>NDCG@1</th>
<th>NDCG@5</th>
<th>NDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>LambdaMART (RankLib)</td>
<td>45.35</td>
<td>44.59</td>
<td>46.46</td>
</tr>
<tr>
<td>LambdaMART (lightGBM)</td>
<td><strong>50.75</strong></td>
<td>49.66</td>
<td>51.48</td>
</tr>
<tr>
<td>LambdaMART + DLCM [1]</td>
<td>46.30</td>
<td>45.00</td>
<td>46.90</td>
</tr>
<tr>
<td>GSF(m=64) with Softmax loss [2]</td>
<td>44.21</td>
<td>44.46</td>
<td>46.77</td>
</tr>
<tr>
<td>FFNN with E[ApproxNDCG] [4]</td>
<td>49.51</td>
<td>48.20</td>
<td>49.96</td>
</tr>
<tr>
<td>TransformerEncoder w/o position [16]</td>
<td>48.58</td>
<td>48.04</td>
<td>50.15</td>
</tr>
<tr>
<td>attn-DIN with Softmax Loss</td>
<td>50.05</td>
<td><strong>50.14^\Delta</strong></td>
<td><strong>52.18^\Delta</strong></td>
</tr>
</tbody>
</table>
Recap

- **TF-Ranking is a deep learning library for LTR**
  - Commonly used ranking losses and metrics
  - Well suited for handling sparse features like text
  - Scales to massive datasets

- **New state-of-the-art solutions for industry applications**
  - TFR-BERT
  - Neural GAM
  - Document Interaction Network (coming soon)
Questions

Try it out: git.io/tf-ranking-demo