TF-Ranking

Neural Learning to Rank using TensorFlow

SIGIR 2019

Rama Kumar Pasumarthi
Sebastian Bruch
Michael Bendersky
Xuanhui Wang

Google Research
Talk Outline

1. Motivation
2. Library overview
3. Empirical results
4. Hands-on tutorial
Motivation
TensorFlow Ranking

- First announced in Google AI blog, Dec. 5th 2018
- The first deep learning library for learning-to-rank at scale
- Available on Github under tensorflow/ranking
- 1100+ stars, 150+ forks
- Actively maintained & developed by the TF-Ranking team
- Compatible with TensorFlow Ecosystem, e.g., TensorFlow Serving
Example I: Search in Gmail

```
Search for: sigir registration deadline

SIGIR Registration
Sebastian Bruch, Qingyao Ai, me
Jan 14

send us a camera ready copy of the SIGIR paper quickly?
Mingyang Zhang, me, John Foley, Marc Najork
4/19/18

Abstract and title
me, Qingyao Ai, Sebastian Bruch
Jan 16

SIGIR2018 notification for short paper 794
SIGIR2018, me
4/11/18

ACM Rights Management: SIGIR ’18 - sp794
John Foley, me, Mingyang Zhang, Marc Najork
4/26/18

WWW 2018 notification for paper 966
John Foley, me, Mingyang Zhang, Marc Najork
2/4/18
```
Example II: Recommendation in Google Drive
General Problem Statement

Problem Learning a scoring function $f^*$ to sort a list of examples

- Input: List of examples (with Context)
- Output: Scoring function $f^*$ that produces the most optimal example ordering
  - Can be parameterized by linear functions, SVM, GBRTs, Neural Networks

Formally

$$
\psi = (x, y) \in X^n \times \mathbb{R}^n
$$

Training sample with relevance labels

$$
\mathcal{L}(f) = \frac{1}{|\psi|} \sum_{(x, y) \in \psi} \ell(y, f(x))
$$

Choose $f^*$ to minimize empirical loss
Most generally

The perfect ranking

64 neurons

128 neurons

256 neurons

Document List
Pointwise Loss (Classification/Regression)

Probability that A is clicked

64 neurons

128 neurons

256 neurons

Doc A
Pairwise Loss

Probability that A is better than B

Doc A

64 neurons

128 neurons

256 neurons

Doc B

64 neurons

128 neurons

256 neurons
Listwise Loss

Probability of the permutation $A > B > C$ (Plackett-Luce model)

- Doc A: 256 neurons - 128 neurons - 64 neurons
- Doc B: 256 neurons - 128 neurons - 64 neurons
- Doc C: 256 neurons - 128 neurons - 64 neurons
Overview
A unified deep learning library for learning-to-rank.
Supported Components

- Supports multivariate scoring functions
- Supports pointwise/pairwise/listwise losses
- Supports popular ranking metrics
  - Mean Reciprocal Rank (MRR)
  - Normalized Discounted Cumulative Gain (NDCG)
- Weighted losses and metrics to support unbiased learning-to-rank
- Supports sparse/embedding features
Supported Scoring Functions

- **Univariate** - scoring function $f(x)$ scores each document separately (most existing LTR methods)

- **Bivariate** - scoring function $f(x_1, x_2)$ scores a pair of documents

- **Multivariate** - scoring functions $f(x_1, \ldots, x_m)$ jointly scores a group of $m$ documents
Groupwise Multivariate Scoring Functions

\[ f(x) \]

\[ \sum \]

\[ \sum \]

\[ \sum \]

\[ g([x_3, x_2]) \]

\[ [x_1, x_2], [x_1, x_3], [x_2, x_1], [x_2, x_3], [x_3, x_1], [x_3, x_2] \]

\[ \mathcal{X} \{x_1, x_2, x_3\} \]
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)

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

"An Analysis of the Softmax Cross Entropy Loss for Learning-to-Rank with Binary Relevance"  
Bruch et al., ICTIR 2019 (to appear)
ApproxNDCG - Ranking Metric Approximation

\[
DCG(\pi_f, y) = \sum_{j=1}^{n} \frac{2y_j - 1}{\log_2(1 + \pi_f(j))}
\]

\[
\pi_f(i) \triangleq 1 + \sum_{j \neq i} \mathbb{I}_{f(x)|i < f(x)|j}
\]

\[
\mathbb{I}_{s < t} = \mathbb{I}_{t-s > 0} \approx \sigma(t - s) \triangleq \frac{1}{1 + e^{-\alpha(t-s)}}
\]

"A general approximation framework for direct optimization of information retrieval measures"
Qin et al., Information Retrieval, 2010

"Revisiting Approximate Metric Optimization in the Age of Deep Neural Networks"
Bruch et al., SIGIR 2019
Empirical Results
## Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th># queries</th>
<th>Type</th>
<th>Domain</th>
<th>Feature Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSLR-Web30k, Yahoo! LTRC</td>
<td>~30K</td>
<td>Public</td>
<td>Search</td>
<td>dense features</td>
</tr>
<tr>
<td>MS-Marco</td>
<td>~800K</td>
<td>Public</td>
<td>Q&amp;A</td>
<td>sparse features</td>
</tr>
<tr>
<td>Quick Access</td>
<td>~30M</td>
<td>Internal</td>
<td>Recommendation</td>
<td>dense features</td>
</tr>
<tr>
<td>Gmail Search</td>
<td>~300M</td>
<td>Internal</td>
<td>Search</td>
<td>dense features, sparse features</td>
</tr>
</tbody>
</table>
Preliminary Results on MS-Marco

● TF-Ranking enables faster iterations over ideas to build ranking-appropriate modules
● An early attempt is illustrated to the right
  ○ Trained with Softmax Cross Entropy (ListNet) loss, it achieves MRR of .244 on the (held-out) “dev” set.
    ■ [Official Baseline] BM25 -- .167
    ■ [Official Baseline] Duet V2 -- .243
    ■ Best non-BERT result -- .318
Gmail Search

Model performance with various loss functions

<table>
<thead>
<tr>
<th>Gmail Search</th>
<th>ΔMRR</th>
<th>ΔARP</th>
<th>ΔNDCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sigmoid Cross Entropy (Pointwise)</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Logistic Loss (Pairwise)</td>
<td>+1.52</td>
<td>+1.64</td>
<td>+1.00</td>
</tr>
<tr>
<td>Softmax Cross Entropy (Listwise)</td>
<td><strong>+1.80</strong></td>
<td><strong>+1.88</strong></td>
<td>+1.57</td>
</tr>
</tbody>
</table>

"TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank"
Pasumarthi et al., KDD 2019 (to appear)
### Quick Access

**Model performance with various loss functions**

<table>
<thead>
<tr>
<th>Quick Access</th>
<th>ΔMRR</th>
<th>ΔARP</th>
<th>ΔNDCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sigmoid Cross Entropy (Pointwise)</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Logistic Loss (Pairwise)</td>
<td>+0.70</td>
<td>+1.86</td>
<td>+0.35</td>
</tr>
<tr>
<td>Softmax Cross Entropy (Listwise)</td>
<td><strong>+1.08</strong></td>
<td><strong>+1.88</strong></td>
<td><strong>+1.05</strong></td>
</tr>
</tbody>
</table>

"TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank"
Pasumarthi et al., KDD 2019 (to appear)
Gmail Search: Incorporating Sparse Features

Model performance with various loss functions

<table>
<thead>
<tr>
<th>Model</th>
<th>ΔMRR</th>
<th>ΔARP</th>
<th>ΔNDCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gmail Search</td>
<td>+6.06</td>
<td>+6.87</td>
<td>+3.92</td>
</tr>
<tr>
<td>Sigmoid Cross Entropy (Pointwise)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logistic Loss (Pairwise)</td>
<td>+5.40</td>
<td>+6.25</td>
<td>+3.51</td>
</tr>
<tr>
<td>Softmax Cross Entropy (Listwise)</td>
<td>+5.69</td>
<td>+6.25</td>
<td>+3.70</td>
</tr>
</tbody>
</table>

"TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank"
Pasumarthi et al., KDD 2019 (to appear)
Hands-on Tutorial
Ecosystem

- Feature Transforms
- Scoring Function
- Ranking Head
- datasets
- losses
- metrics
- tf.data
- Layers
- Feature Columns
- Model Builder
- Ranking Building Blocks
- TensorFlow Core
- Python Ops
- C++ Ops
- TensorFlow Distributed Execution Engine
- CPU
- GPU
- Android
- iOS
- ...
TF-Ranking Architecture

Training

Data → Input Reader

Input Reader:
- `input_fn`

Feature Transformation:
- `transform_fn`

Scoring Function:
- `score_fn`

Scores → Ranking Head

Ranking Head

Metrics Builder:
- `make_metrics_fn`

Loss Builder:
- `make_loss_fn`

Labels → Loss Metrics

Serving

raw data → Serving receiver

Serving receiver:
- `input_fn`

Feature Transformation:
- `transform_fn`

Features → Scoring Function

Scoring Function:
- `score_fn`

Scores
Steps to get started

- Go to [git.io/tf-ranking-demo](https://git.io/tf-ranking-demo)
- Open the notebook in colab.
  - Make sure the URL starts with “colab.research.google.com”
- Click “Connect” to connect to a hosted runtime.
  - This is where the code runs, and the files reside.
- Open “Runtime” and select “Run All”
- Scroll down to the section on “Train and evaluate the ranker”, to see the training in execution
git.io/tf-ranking-demo
"Course Homework"

- Try running the colab with a different loss function
  - Use one of the losses listed at: [git.io/tfr-losses](https://git.io/tfr-losses)
  - Advanced: Implement your own custom loss function

- Try running with an additional metric
  - You can use Average Relevance Position, listed at: [git.io/tfr-metrics](https://git.io/tfr-metrics)
  - Advanced: Implement a metric that is a linear combination of two existing metrics

- Explore different neural networks for scoring function
  - Increase the number of layers: when does it start to overfit?

- Try running TF-Ranking on your ranking problem
  - Let us know your experience by filing an [issue](https://github.com) on github!