TF-Ranking

Neural Learning to Rank using TensorFlow

SIGIR 2019

Rama Kumar Pasumarthi
Sebastian Bruch
Michael Bendersky
Xuanhui Wang

Google Research

TensorFlow

Talk Outline

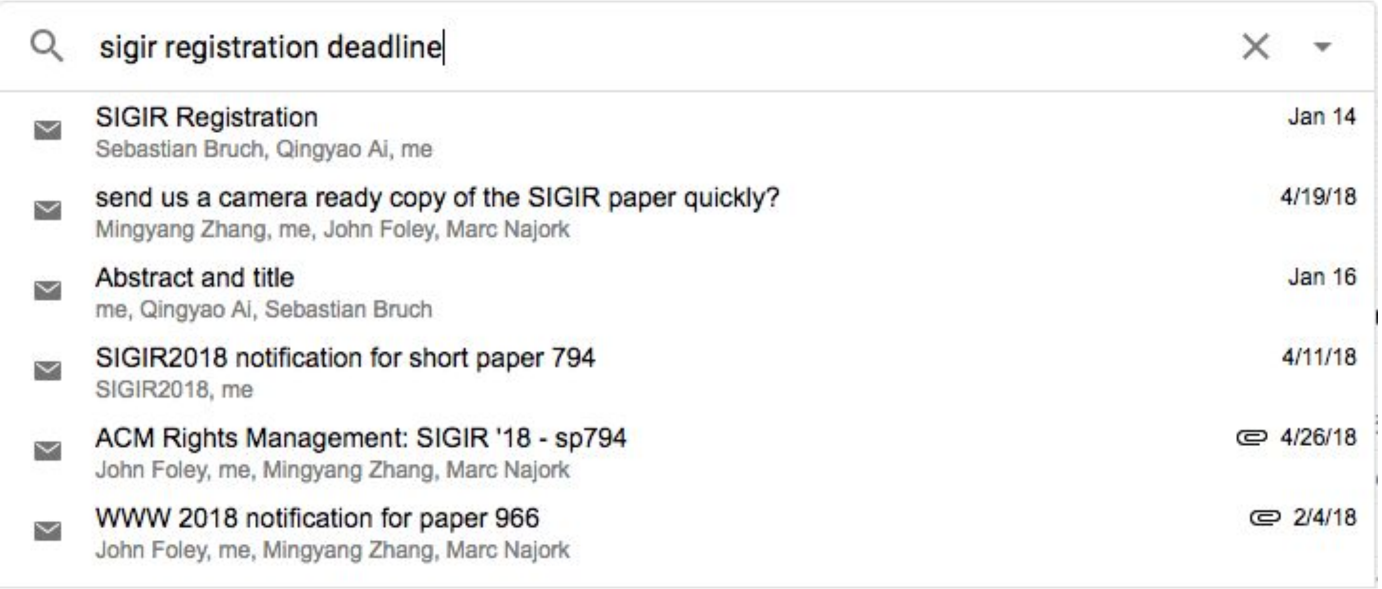
1. **Motivation**
2. **Library overview**
3. **Empirical results**
4. **Hands-on tutorial**

Motivation

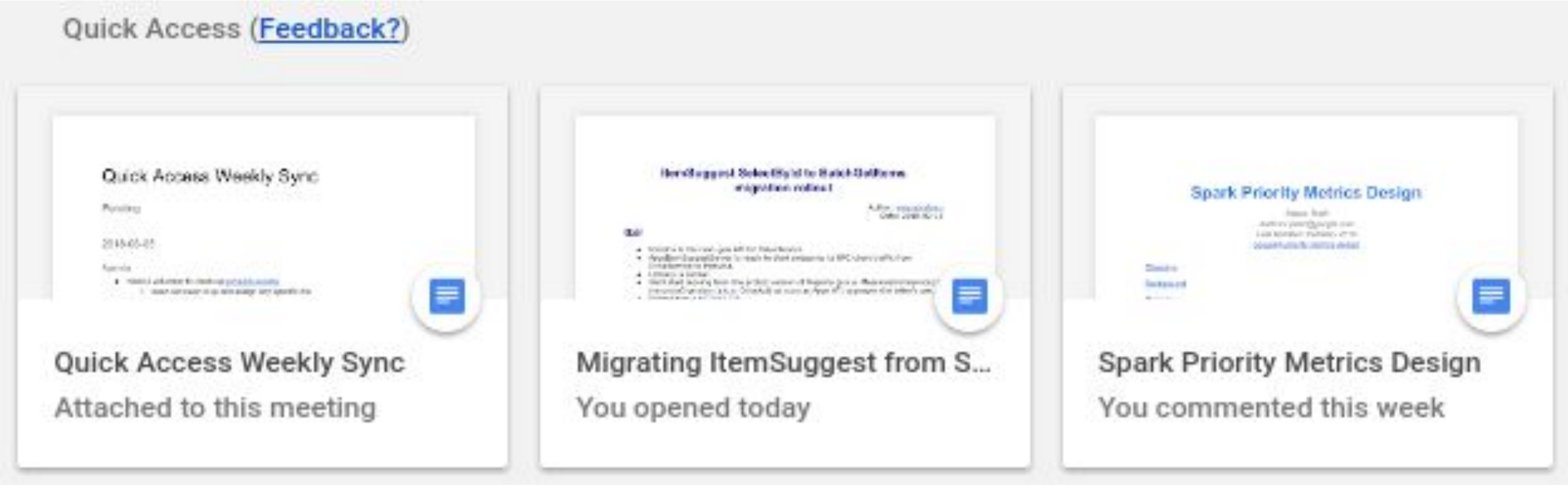
TensorFlow Ranking

- 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



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

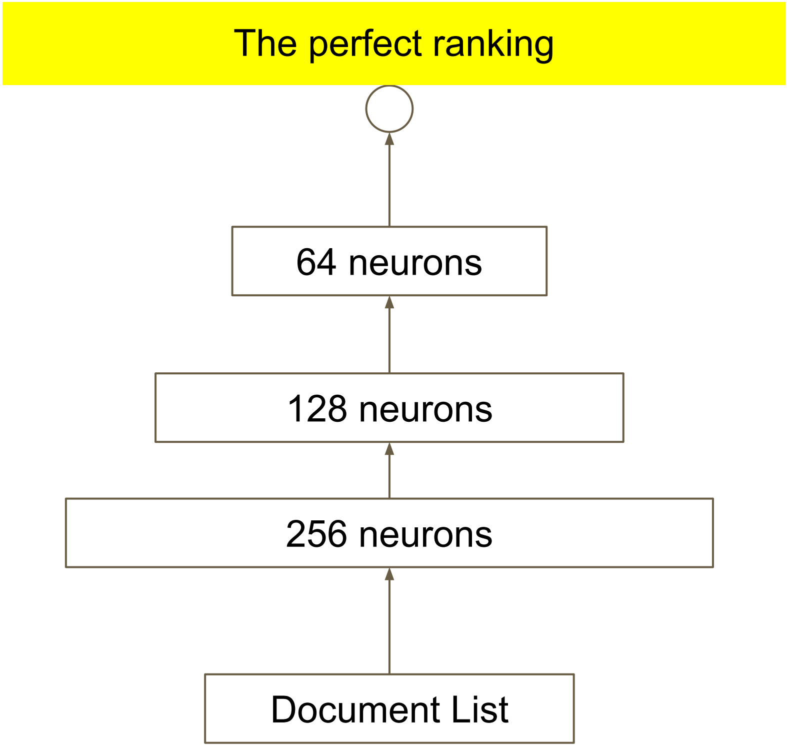
$$\psi = (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X}^n \times \mathbb{R}^n$$

Training sample with relevance labels

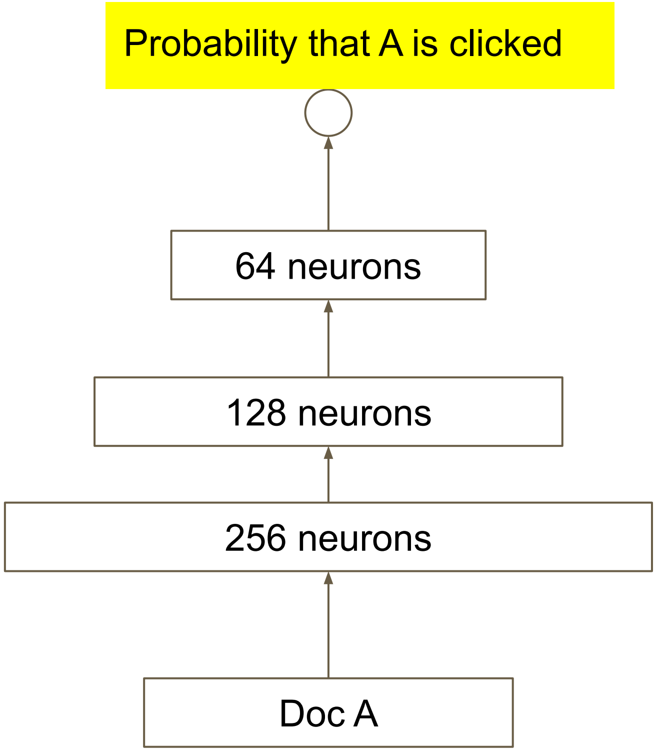
$$\mathcal{L}(f) = \frac{1}{|\Psi|} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in \Psi} \ell(\boldsymbol{y}, f(\boldsymbol{x}))$$

Choose f* to minimize empirical loss

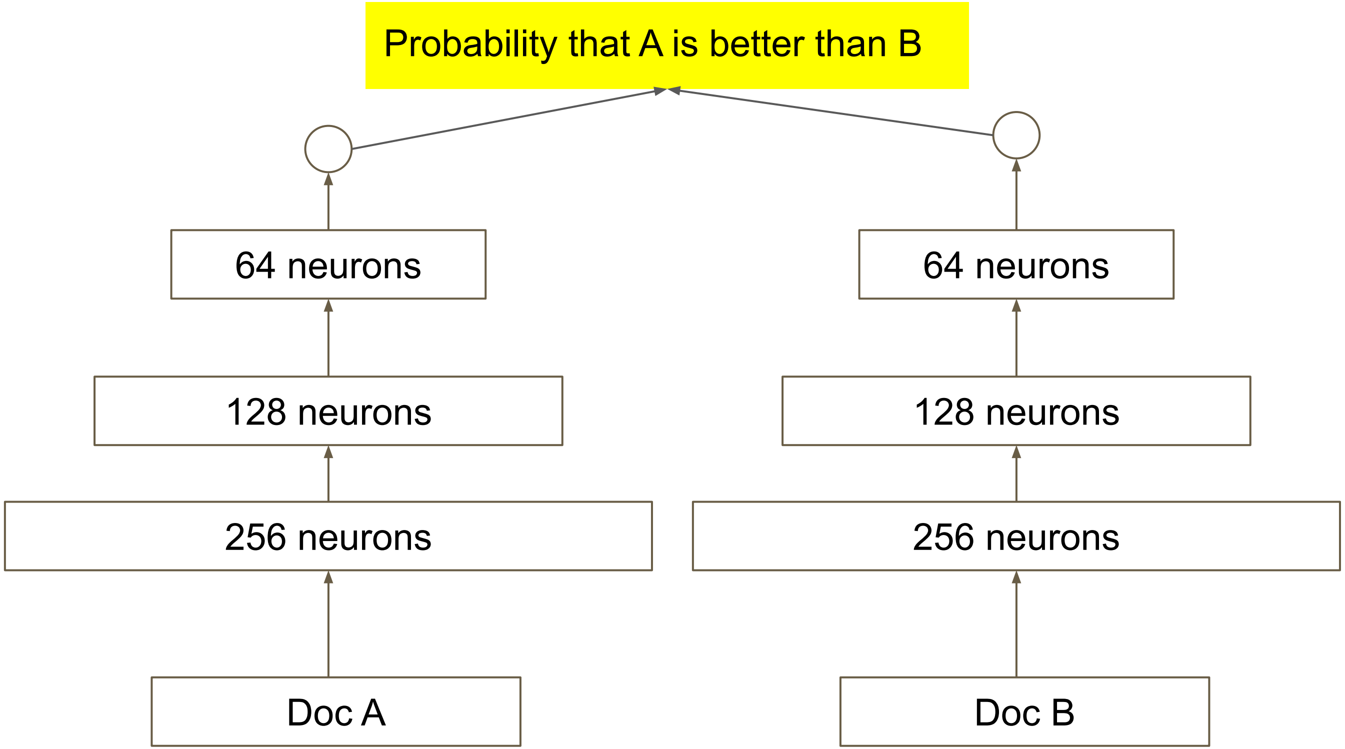
Most generally



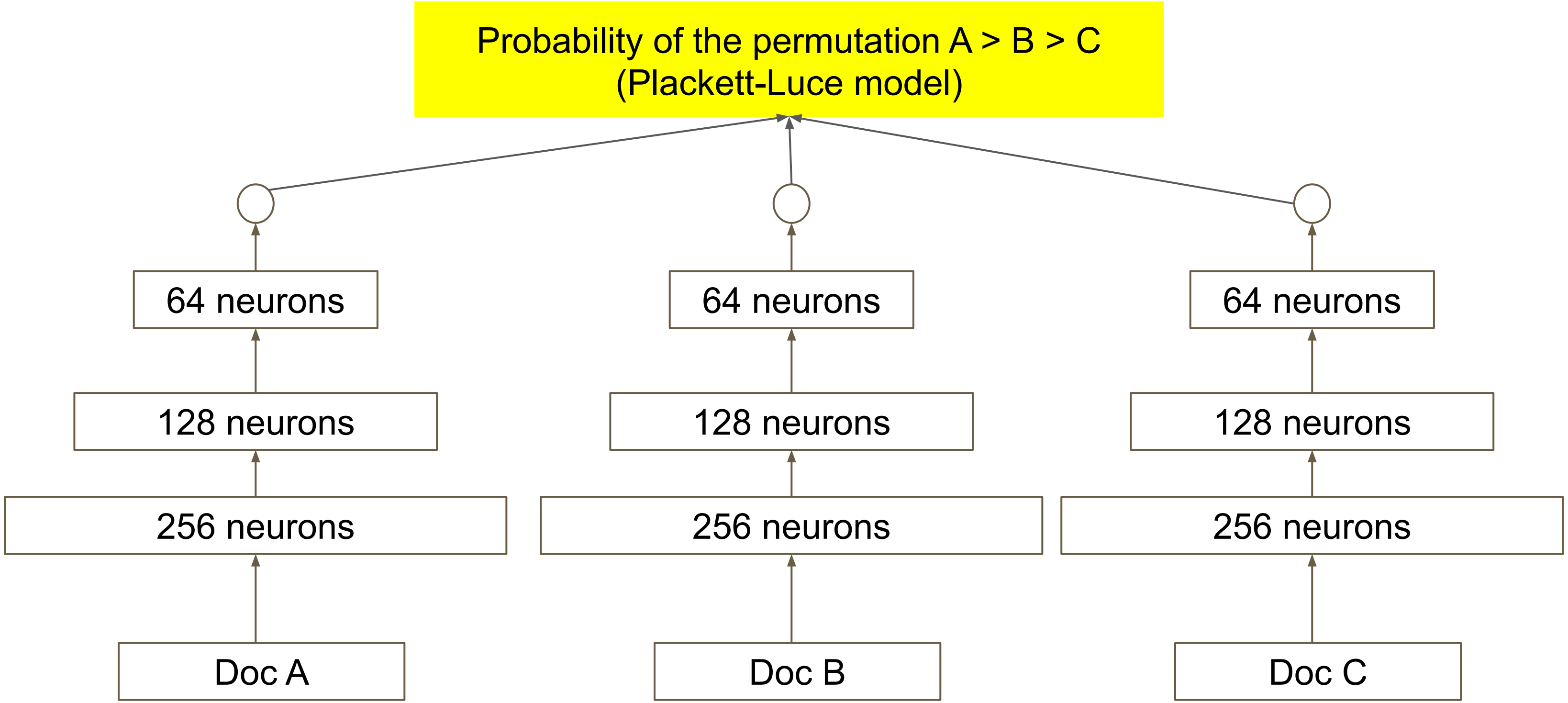
Pointwise Loss (Classification/Regression)



Pairwise Loss



Listwise Loss



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

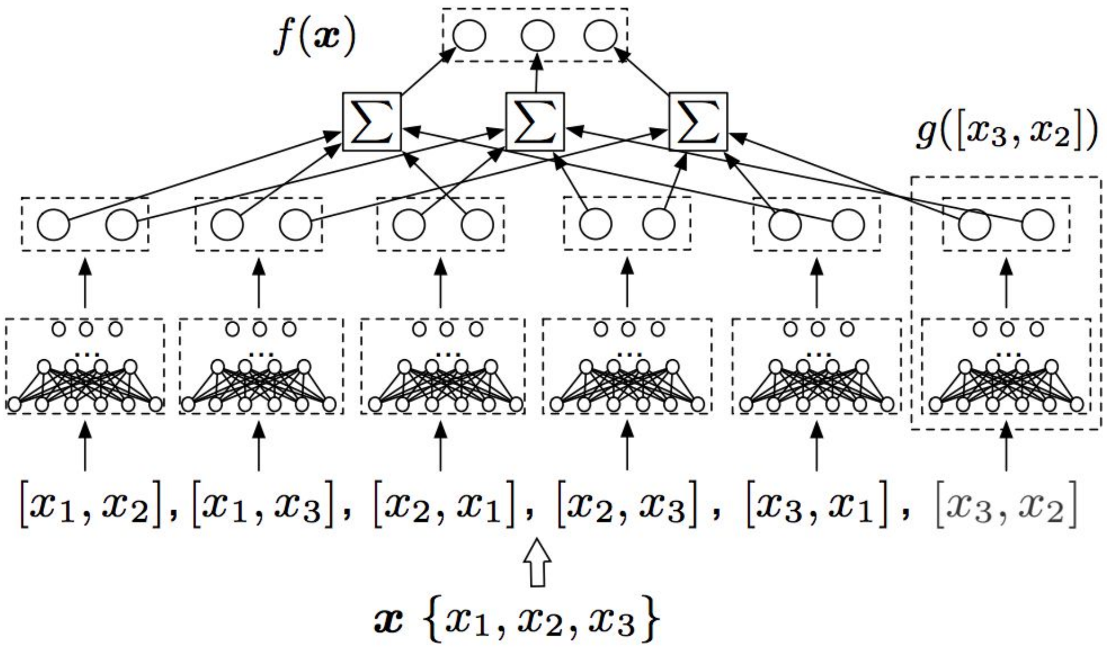
Supported Scoring Functions

- **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



Supported Loss Examples (Binary Labels)

(*Pointwise*) Sigmoid Cross Entropy

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

(Pairwise) Logistic Loss

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

(Listwise) Softmax Loss (aka ListNET)

$$\hat{\ell}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_{j=1}^{n} y_j \log(\frac{\exp(\hat{y}_j)}{\sum_{j=1}^{n}\exp(\hat{y}_j)})$$

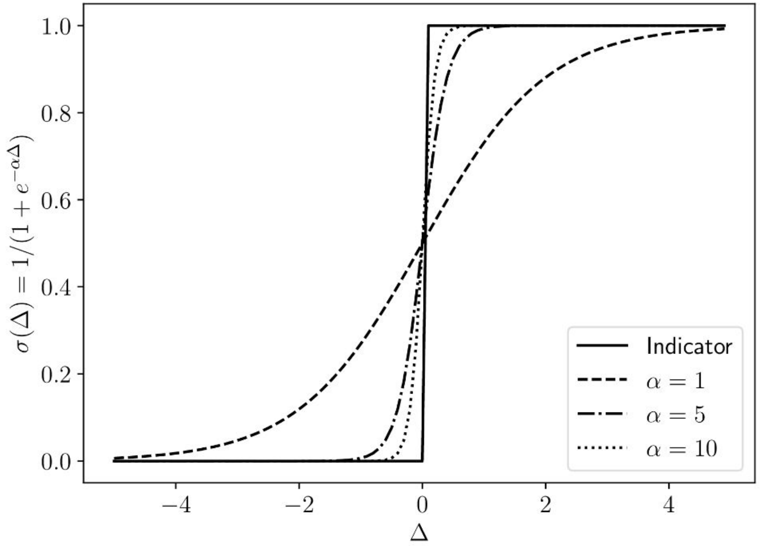
"An Analysis of the Softmax Cross Entropy Loss for Learning-to-Rank with Binary Relevance"
Bruch et al., ICTIR 2019 (to appear)

ApproxNDCG - Ranking Metric Approximation

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

$$\pi_f(i) \triangleq 1 + \sum_{j \neq i} \mathbb{I}_{f(\boldsymbol{x})|_i < f(\boldsymbol{x})|_j}$$

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



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

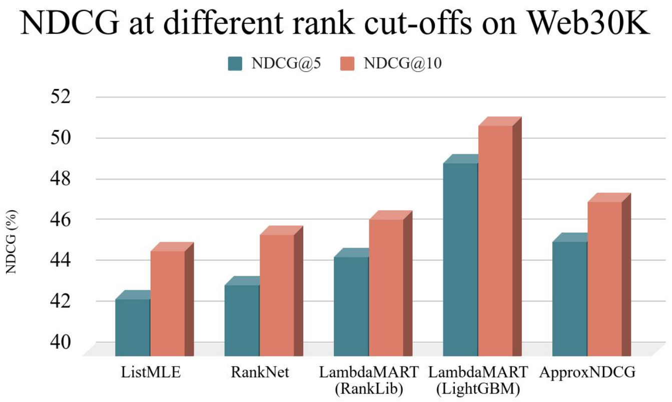
"Revisiting Approximate Metric Optimization in the Age of Deep Neural Networks"
Bruch et al., SIGIR 2019

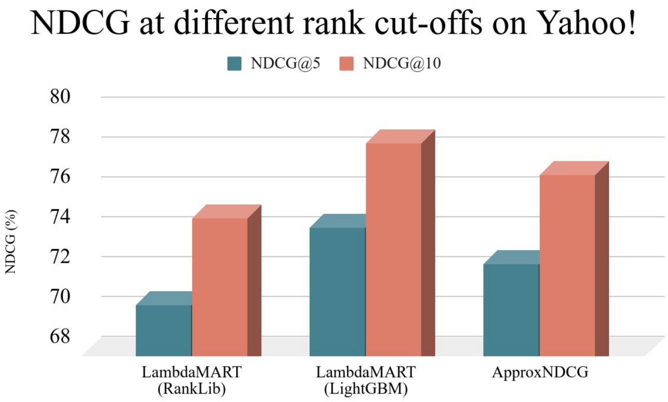
Empirical Results

Datasets

Dataset	# queries			
MSLR-Web30k, Yahoo! LTRC	~30K	Public	Search	dense features
MS-Marco	~800K	Public	Q&A	sparse features
Quick Access	~30M	Internal	Recommendation	dense features
Gmail Search	~300M	Internal	Search	dense features sparse features

MSLR-Web30k and Yahoo! LTRC

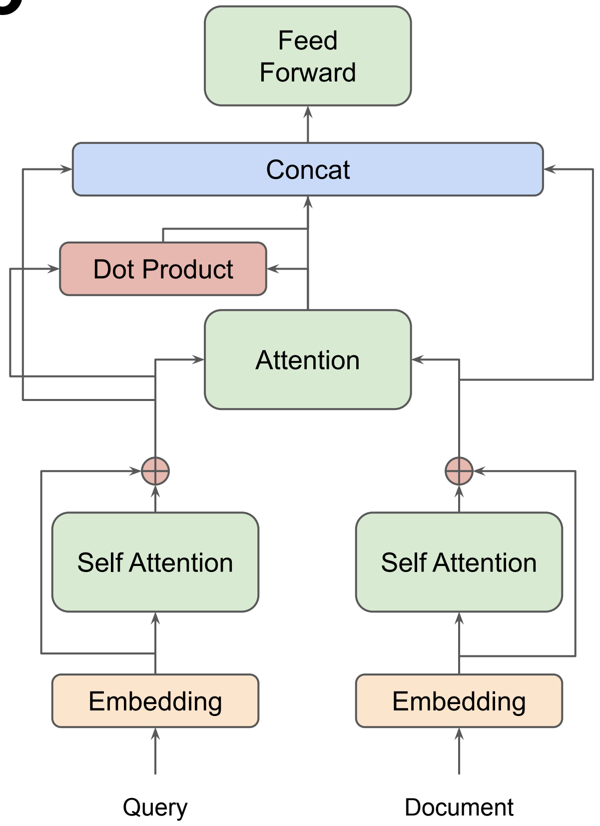




"Revisiting Approximate Metric Optimization in the Age of Deep Neural Networks"
Bruch et al., SIGIR 2019

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

Gmail Search	ΔMRR	ΔARP	ΔNDCG
Sigmoid Cross Entropy (Pointwise)	–	–	–
Logistic Loss (Pairwise)	+1.52	+1.64	+1.00
Softmax Cross Entropy (Listwise)	**+1.80**	**+1.88**	**+1.57**

Quick Access

Model performance with various loss functions

Quick Access	ΔMRR	ΔARP	ΔNDCG
Sigmoid Cross Entropy (Pointwise)	–	–	–
Logistic Loss (Pairwise)	+0.70	+1.86	+0.35
Softmax Cross Entropy (Listwise)	**+1.08**	**+1.88**	**+1.05**

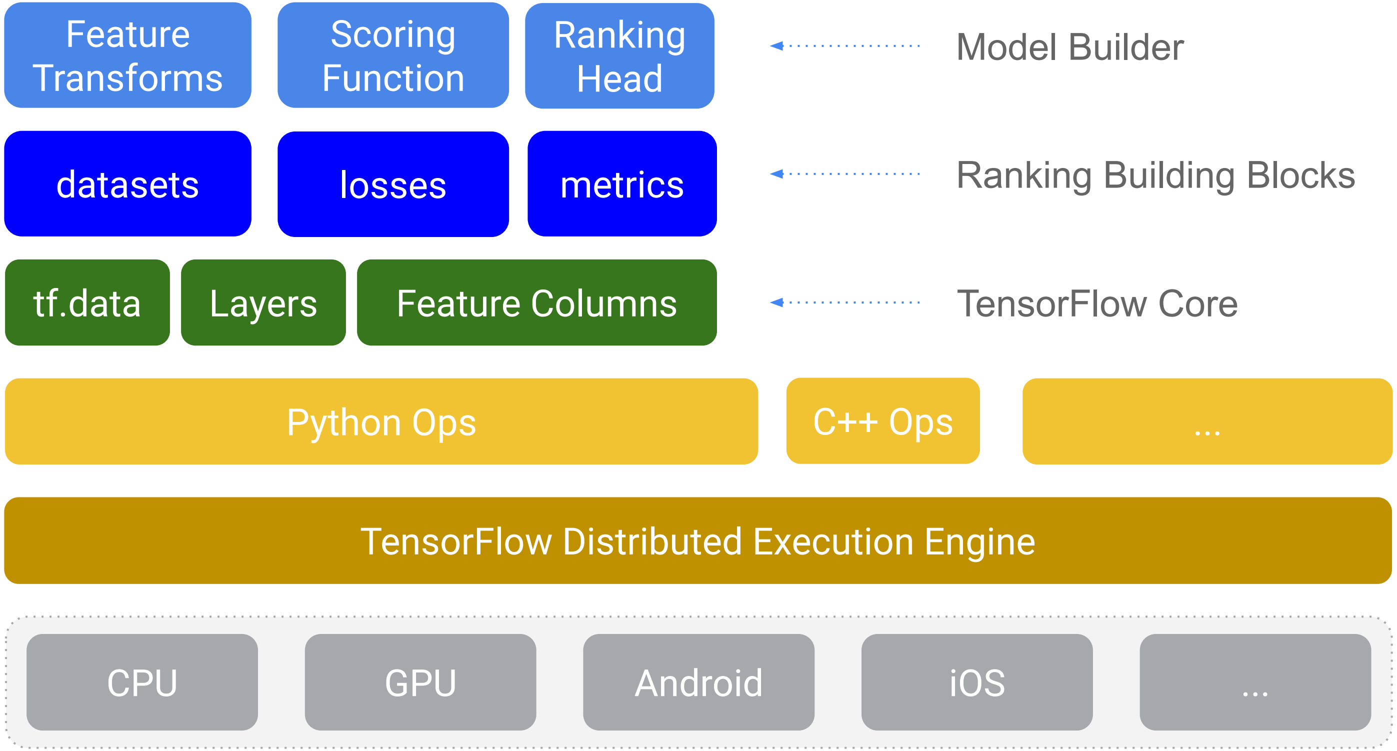
Gmail Search: Incorporating Sparse Features

Model performance with various loss functions

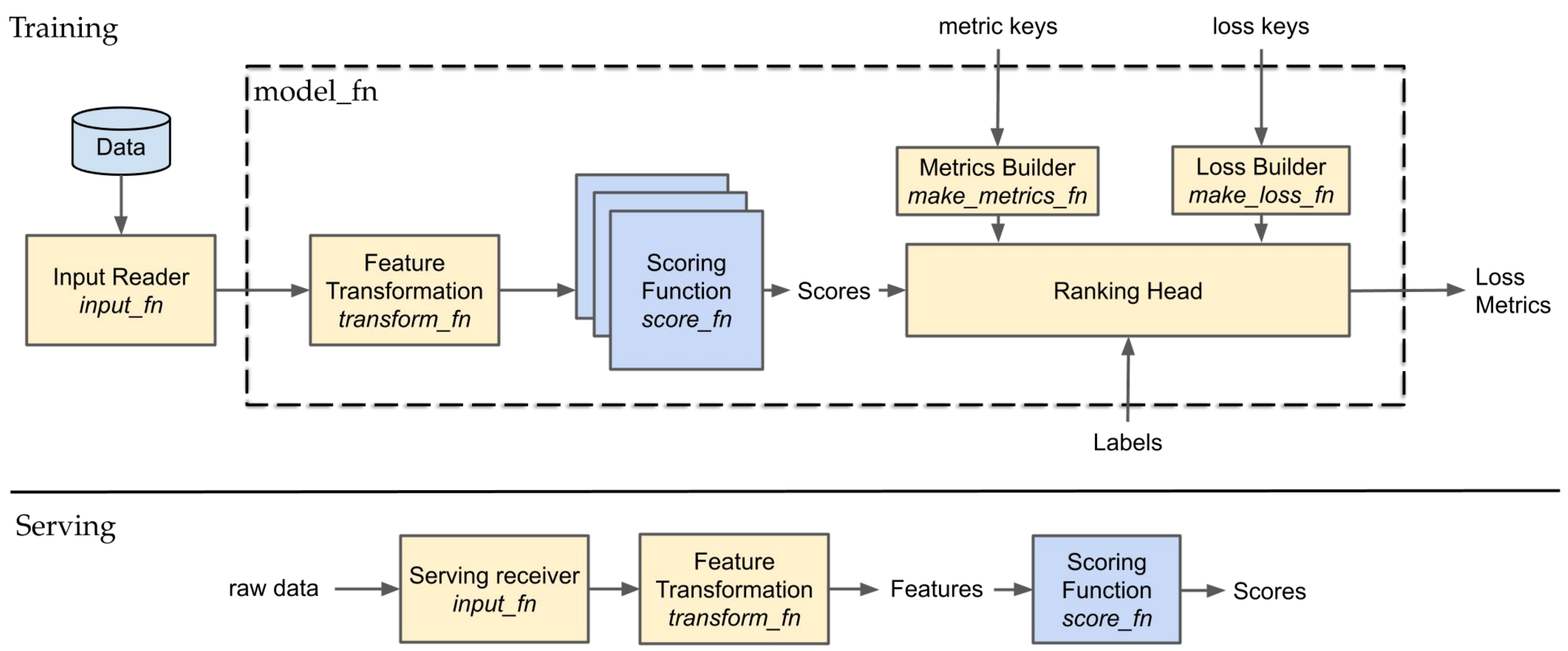
Gmail Search	ΔMRR	ΔARP	ΔNDCG
Sigmoid Cross Entropy (Pointwise)	+6.06	+6.87	+3.92
Logistic Loss (Pairwise)	+5.40	+6.25	+3.51
Softmax Cross Entropy (Listwise)	+5.69	+6.25	+3.70

Hands-on Tutorial

Ecosystem


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

TF-Ranking Architecture



Steps to get started

- Go to git.io/tf-ranking-demo
- Open the notebook in colaboratory
    - 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
  - 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
  - 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 on github!