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
    - ...

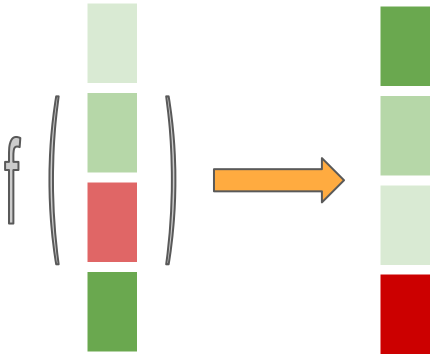
State of the Art on Public Benchmarks

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

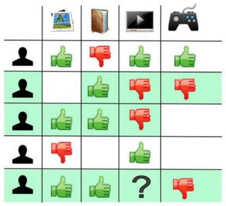
- TREC-COVID19
  - No. 1 in round 4 for 4 out 5 metrics.
  - No. 1 in round 5 for all 5 metrics.

ndcg@20	P@20	rbp_p5	bpref	map
0.8496	0.8760	0.9197	0.6372	0.4718
0.8490	0.8690	0.9399	0.6378	0.4731
0.8311	0.8460	0.9361	0.5330	0.3922
0.8304	0.8380	0.9370	0.5280	0.3875

Learning-to-Rank (LTR)



Search



Recommendation

Question Answering

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

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

Training sample with relevance labels

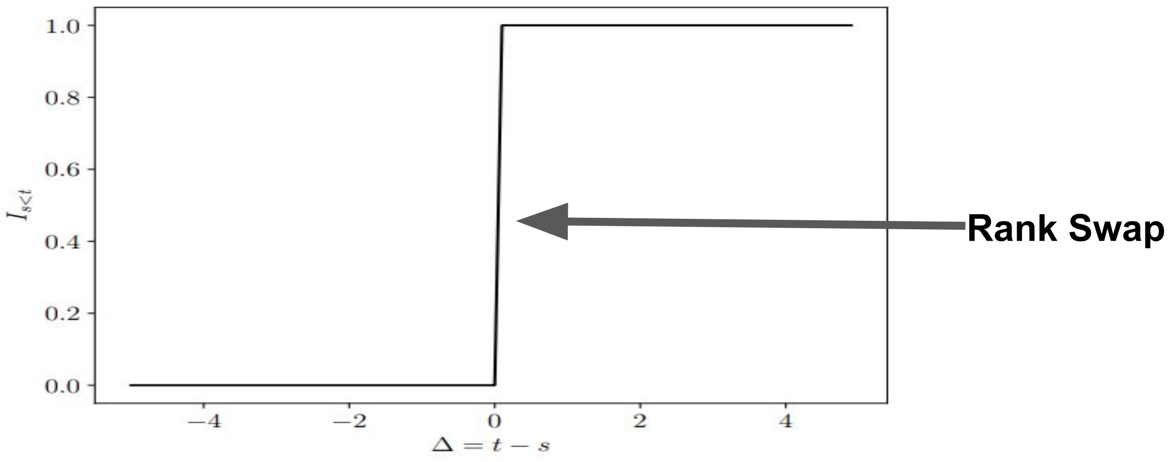
$$\mathcal{L}(f) = \frac{1}{|\Psi|} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in \Psi} \ell(\boldsymbol{y}, f(\boldsymbol{x})),$$

Choose f* to minimize empirical loss

Ranking Metrics

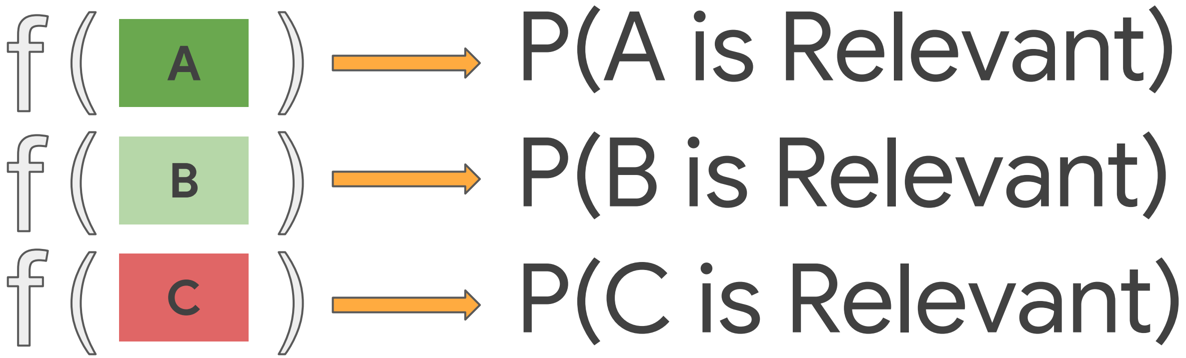
Standard ranking metrics are either **discontinuous** or **flat** everywhere

- Cannot be directly optimized with gradient descent



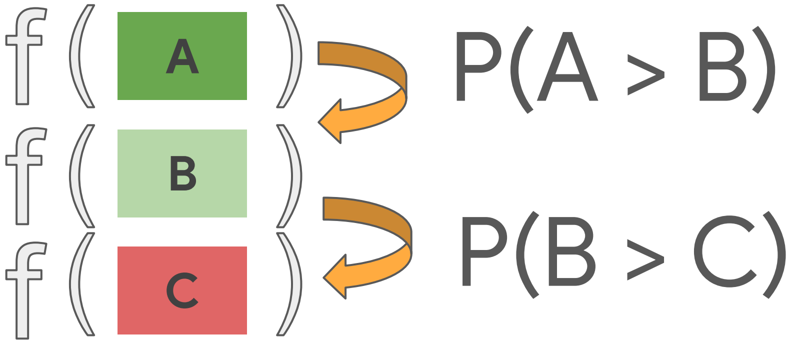
Pointwise LTR methods

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



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)

ELWC: ExampleListWithContext

q_1

```
{
    key: "query_tokens"
    value: ["this", "is", "a", "query"]
}
```

doc list

$d_{1,1}$

```
{
    key: "document_tokens"
    value: ["relevant", "answer"],
    key: "relevance"
    value: 1
}
```

$d_{1,2}$

```
{
    key: "document_tokens"
    value: ["irrelevant", "data"],
    key: "relevance"
    value: 0
}
```

$d_{1,3}$

```
{
    key: "document_tokens"
    value: ["very", "relevant"],
    key: "relevance"
    value: 2
}
```

q_2

```
{
    key: "query_tokens"
    value: ["another", "query"]
}
```

doc list

$d_{2,1}$

```
{
    key: "document_tokens"
    value: ["irrelevant", "answer"],
    key: "relevance"
    value: 0
}
```

$d_{2,2}$

```
{
    key: "document_tokens"
    value: ["relevant"],
    key: "relevance"
    value: 1
}
```

- 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

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

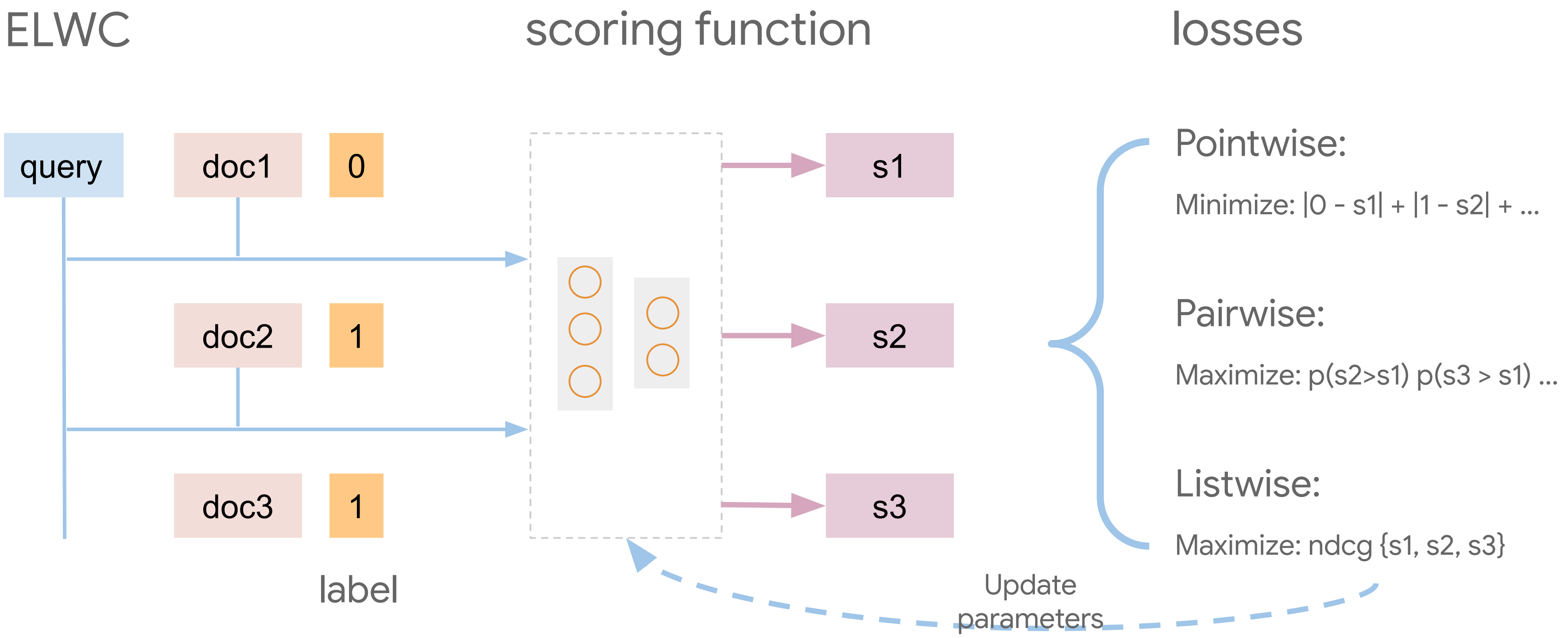
(Pairwise) Logistic Loss

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

(Listwise) Softmax Loss (aka ListNET)

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

TF-Ranking - How it works



New developments in TF-Ranking

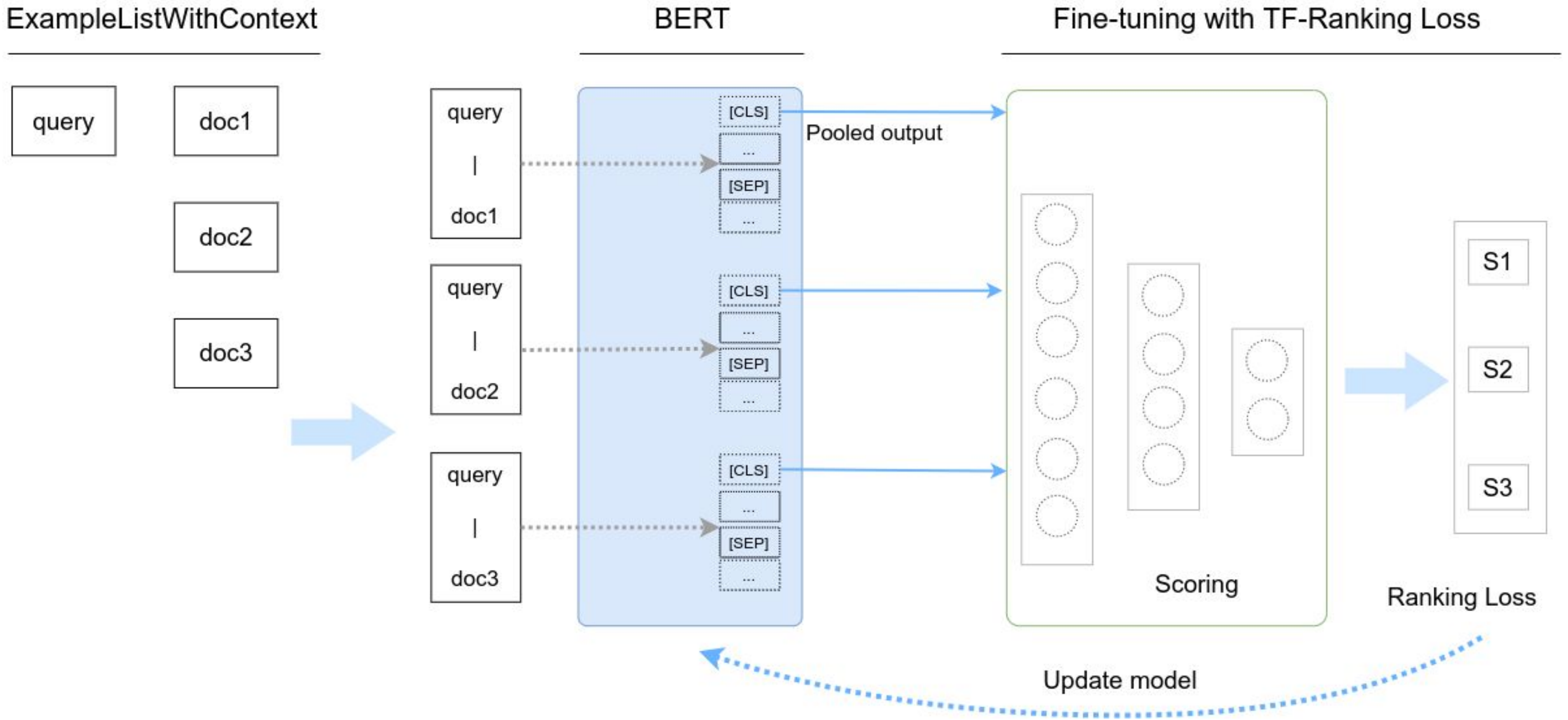
New developments

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

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

3. **Document Interaction Networks**
    - Modeling cross-document interactions

TFR-BERT



[Han et al. arXiv] Learning-to-Rank with BERT in TF-Ranking.

BERT with Ranking Loss

- The model is fine-tuned by "**softmax loss**":

$$\ell_q = \sum_{d \in \mathcal{C}} \frac{y_d}{\sum_{d' \in \mathcal{C}} y_{d'}} \log \left(\frac{\exp(\mathrm{sc}_{\mathrm{BERT}}(d))}{\sum_{d' \in \mathcal{C}} \exp(\mathrm{sc}_{\mathrm{BERT}}(d'))} \right)$$

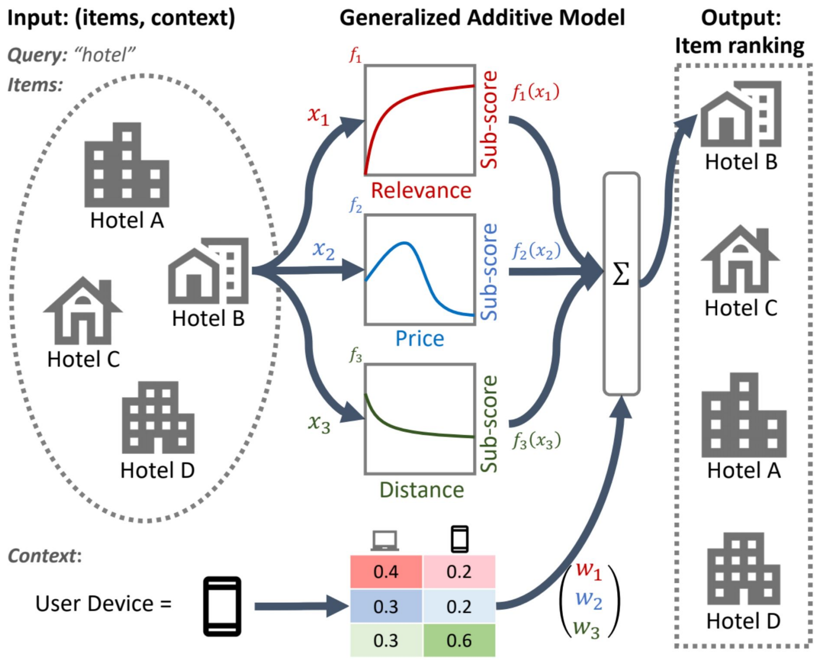
where y_d is the ground-truth label

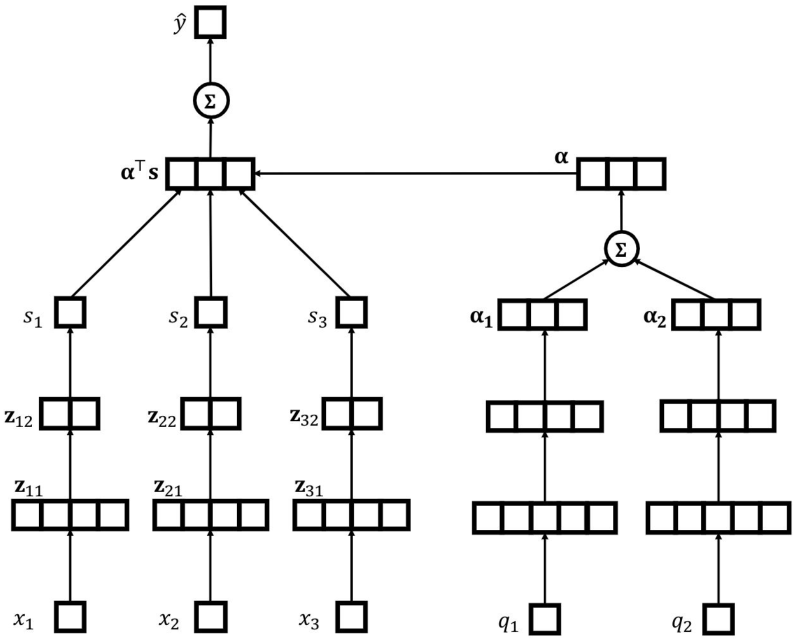
- The loss function considers the other documents in the same list
    - Better ranking performances compared to **sigmoid cross-entropy loss**

Sigmoid CE	37.16
Pairwise log-loss	37.18
Softmax Loss	37.82
Best Ensemble model	**38.77**

Results on MS-Marco passage re-ranking

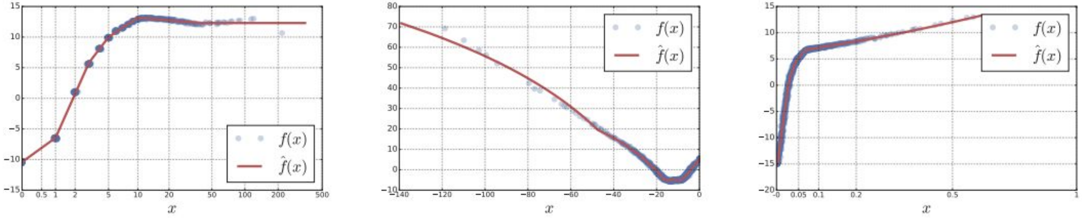
Interpretable LTR models: Neural GAM





[Zhuang et al. WSDM2021] Interpretable Ranking with Generalized Additive Models.

Capabilities

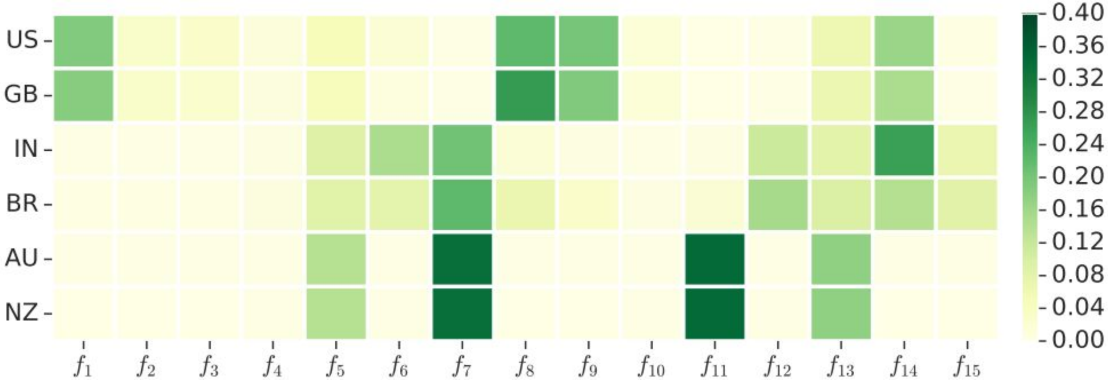


(a) min of term frequency (whole document)

(b) LMIR.JM (body)

(c) sum of stream length normalized term frequency (whole document)

(a) Distilling sub-models as piecewise curves



(b) Measuring the effect of context features

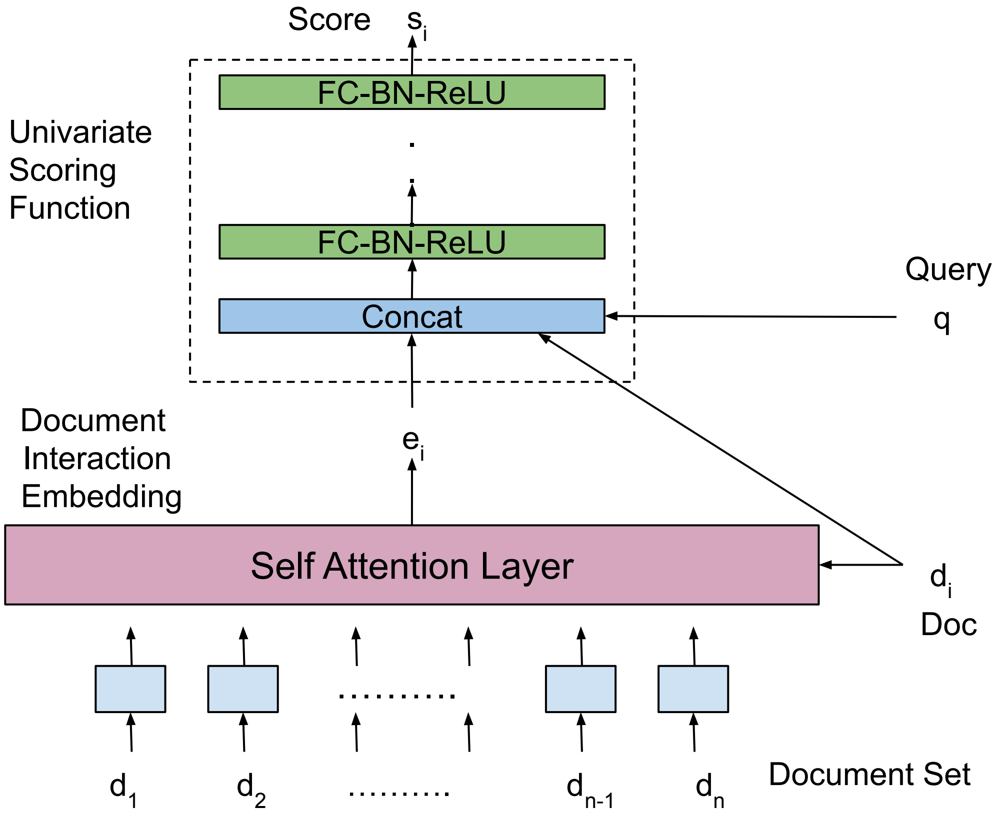
Performance

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

Data set	Method	$NDCG_1$	$NDCG_5$	$NDCG_{10}$
YAHOO	Tree GAM	67.61	69.46	73.89
	Neural GAM	67.63	69.62	73.98
	Tree RankGAM	69.12	71.03	75.04
	Neural RankGAM	**69.36**	**71.32**	**75.33***
WEB30K	Tree GAM	29.79	32.79	35.96
	Neural GAM	30.59	33.55	36.54
	Tree RankGAM	41.90	42.04	44.37
	Neural RankGAM	**44.31***	**43.29***	**45.09***
CWS	Tree GAM	19.74	32.91	36.72
	Neural GAM	20.09	34.01	38.60
	Tree RankGAM	20.16	35.06	39.27
	Neural RankGAM	20.35	34.94	38.93
	Neural RankGAM+	**24.43***	**39.88***	**42.84***

[Pasumarthi et al. ICTIR2020]
Permutation Equivariant Document Interaction Network for Neural Learning to Rank

Document Interaction Network



Experiments on Web30K Benchmark

Method	NDCG@1	NDCG@5	NDCG@10
LambdaMART (RankLib)	45.35	44.59	46.46
LambdaMART (lightGBM)	**50.75**	49.66	51.48
LambdaMART + DLCM [1]	46.30	45.00	46.90
GSF(m=64) with Softmax loss [2]	44.21	44.46	46.77
FFNN with E[ApproxNDCG] [4]	49.51	48.20	49.96
TransformerEncoder w/o position [16]	48.58	48.04	50.15
attn-DIN with Softmax Loss	50.05	**50.14**$^{\triangle}$	**52.18**$^{\triangle}$

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**