Learning to rank with the Transformer

Tomasz Bartczak
Radosław Białobrzeski

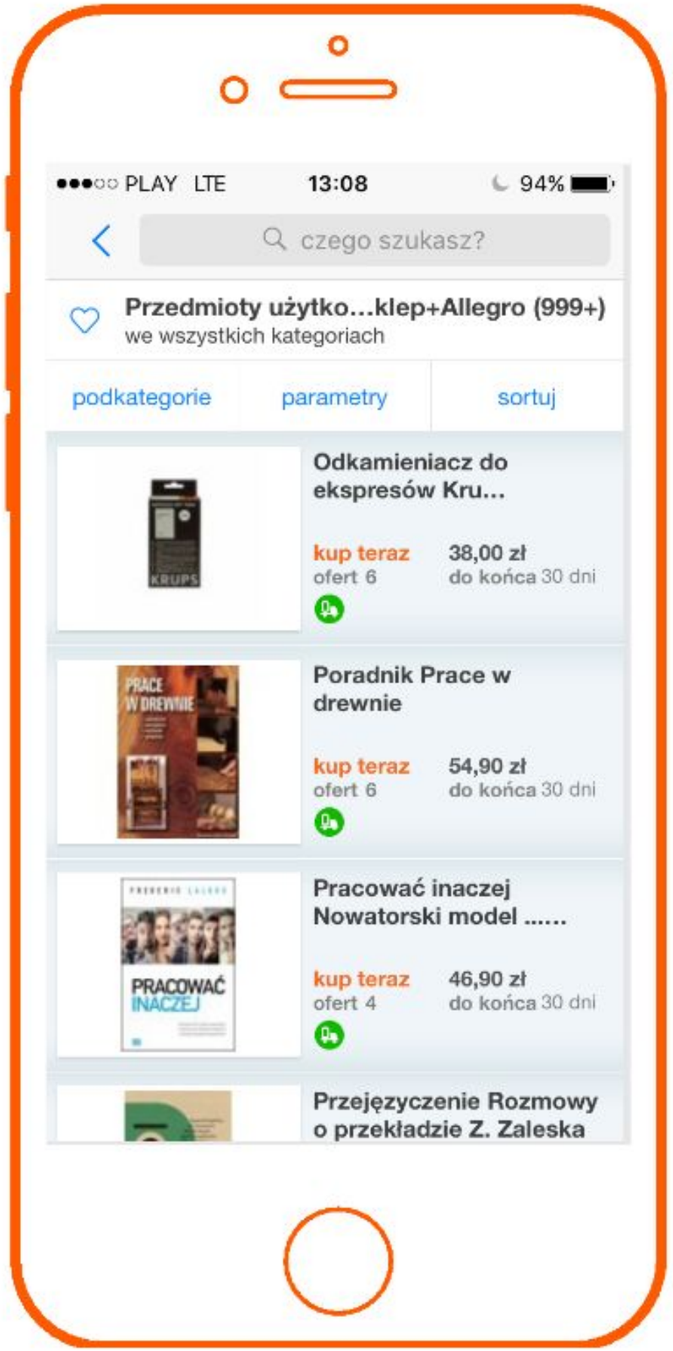
Learning to rank

Learning to rank

- Learning to Rank (LTR) is concerned with optimising the global ordering of a list of items according to their utility to the user
- To frame it as a supervised learning problem, one needs a dataset of query results together with their relevance labels

Learning to rank - datasets

- Item relevance is expressed as:
  - graded relevance (human-annotated, on a scale of, say, 1 to 5) - expensive and small but clean
  - binary relevance (usually, clickthrough logs of a live search engine are used as a proxy for relevance) - cheap and abundant but noisy



Learning to Rank - implicit feedback

[0.19, 0.42, 0.75, ...] skipped

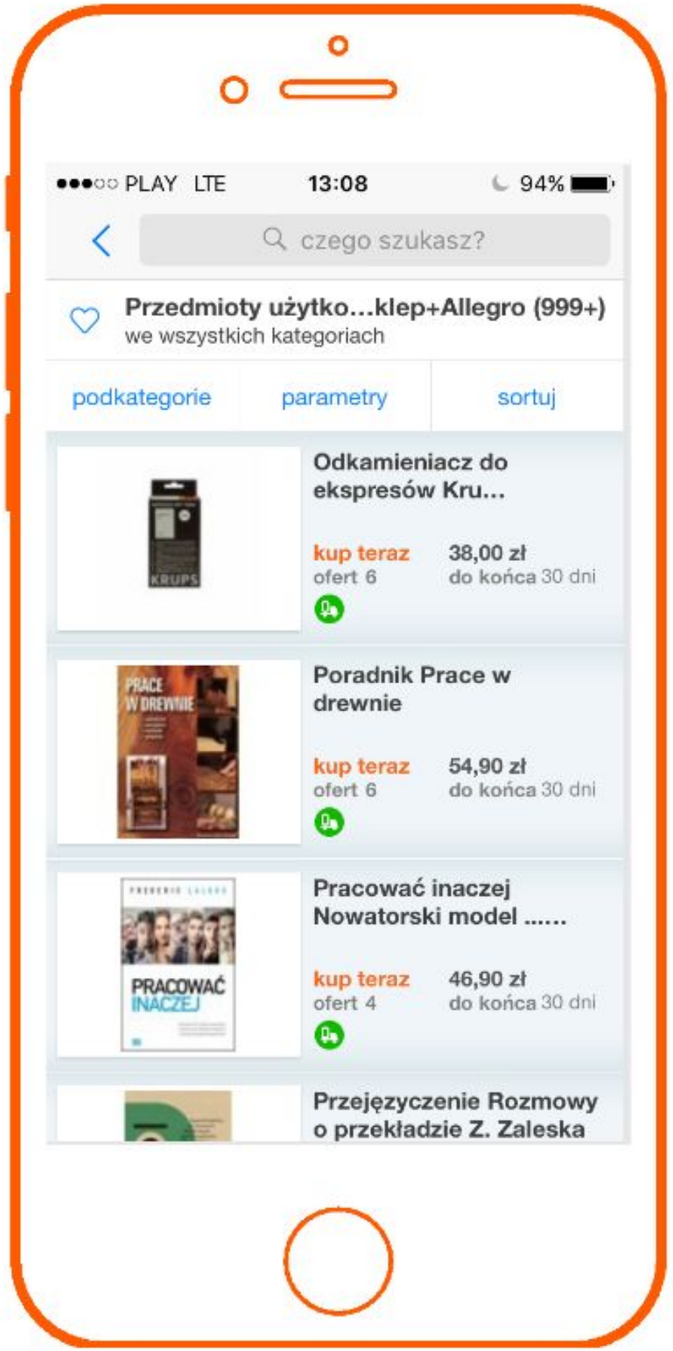
[0.22, -0.35, 0.21, ...] clicked

[0.12, 0.45, 0.11, ...] skipped

Score & sort

- Most LTR algorithms fall into *score & sort* type: using the training data, we actually learn a **scoring** function *f* , which assign scores to items individually.
- Items are then sorted in descending order of the scores
- An example of a different type: Seq2Slate[1]

1. Bello, Irwan, et al. "Seq2slate: Re-ranking and slate optimization with RNNs." *arXiv preprint arXiv:1810.02019* (2018).



score & sort

f([0.19, 0.42, 0.75, ...]) = 0.87

f([0.22, -0.35, 0.21, ...]) = 0.73

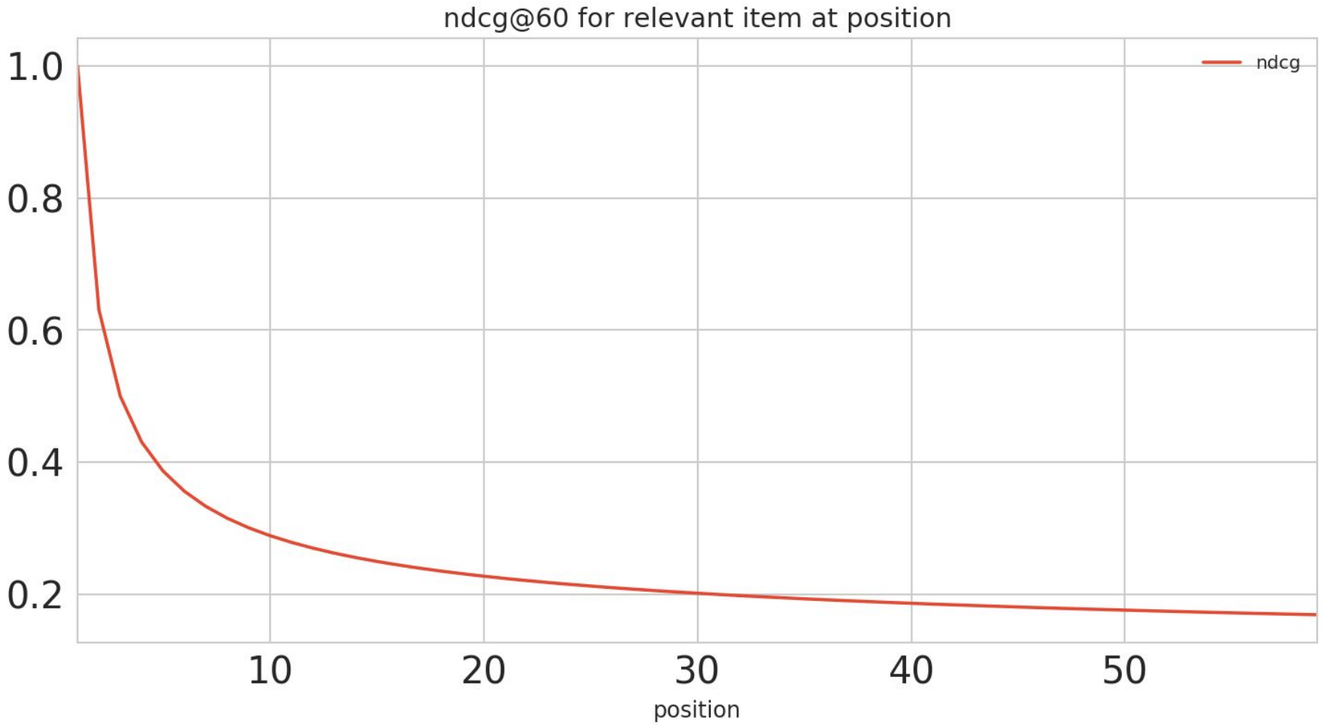
f([0.12, 0.45, 0.11, ...]) = 0.65

Metric: NDCG

A typical metric for ranking is NDCG[1] (Normalized Discounted Cumulative Gain)

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)}$$

$$\mathrm{nDCG_p} = \frac{DCG_p}{IDCG_p},$$



1. Järvelin, Kalervo, and Jaana Kekäläinen. "Cumulated gain-based evaluation of IR techniques." ACM Transactions on Information Systems (TOIS) 20.4 (2002): 422-446.

Training & losses

- Our task is to train a model that maximises NDCG
- We would like to train the model using gradient-based optimiser
- The metric rely on a *sort* operator, which is non differentiable
- There is an ongoing research in the area of approximating NDCG to allow for finding the optimal permutation w.r.t to this metric

Training & losses

- Meanwhile - loss functions for ranking are surrogates for the target metric
  - pointwise - loss defined for a single item - e.g. a classification model
  - pairwise - loss defined for a pair of items - e.g. RankNet[1]
  - listwise - loss defined for a whole list - e.g. ListNet[2]

1. Burges et al. "Learning to Rank Using Gradient Descent." ICML '05 Proceedings of the 22nd International Conference on Machine Learning (2005): 89-96.
2. Xia et al. "Listwise Approach to Learning to Rank: Theory and Algorithm." ICML '08 Proceedings of the 25nd International Conference on Machine Learning (2008): 1192-1199.

Transformer encoding architecture

Transformer

- Transformer architecture was first introduced in *Attention Is All You Need* [1] and has since become the state of the art in NLP
- Processing whole sequence at a time, it eliminates recurrent dependencies, meaning easier parallelization, faster training and inference
- At the same time, it allows for a contextualized and position-aware representation
- Transformer consists of encoder and decoder blocks - we make use only of the encoder blocks

1. Vaswani et al. "Attention Is All You Need." NIPS '17 Proceedings of the 31st International Conference on Neural Information Processing Systems (2017): 6000-6010.

Attention mechanism

- Attention is key to the Transformer architecture
- It provides the contextual awareness when processing a sequence element
- Let Q, K, V be query, key and value matrices[1] - e.g. outputs of a neural encoder transformed in a certain way
- Scaled dot-product attention linearly combines input values for each output position according to softmax of query-key scaled dot products:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

1. This is original paper's terminology. These matrices are somewhat abstract entities. Please do not confuse the query matrix with a search engine query.

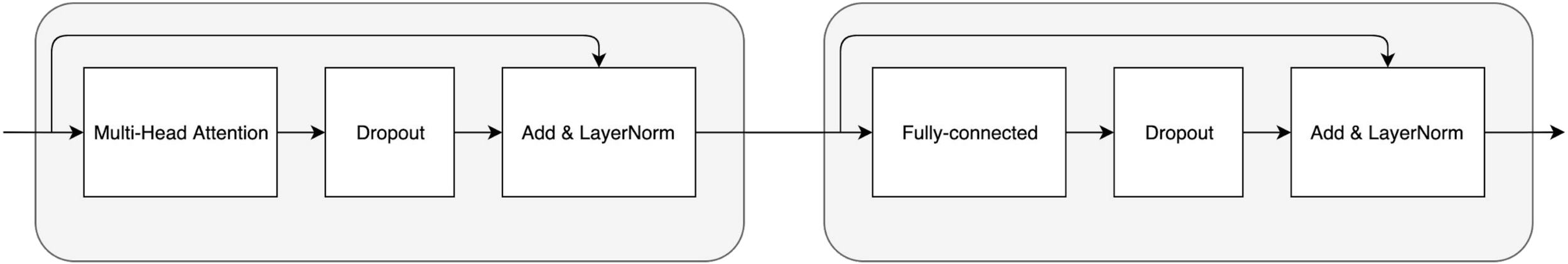
Self-attention

- When the query, key and value matrices are equal or are linear transformations of the same matrix, we obtain **self-attention**
- Applying different sets of learned linear projections to neural encoder's output enables us to use **multi-head attention** - tracking different aspects of the input sequences and combining them into a resourceful representation:

$$\text{MHAttention}(X) = \text{concat}(\text{head}_1, ..., \text{head}_N)$$
$$\text{head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V)$$

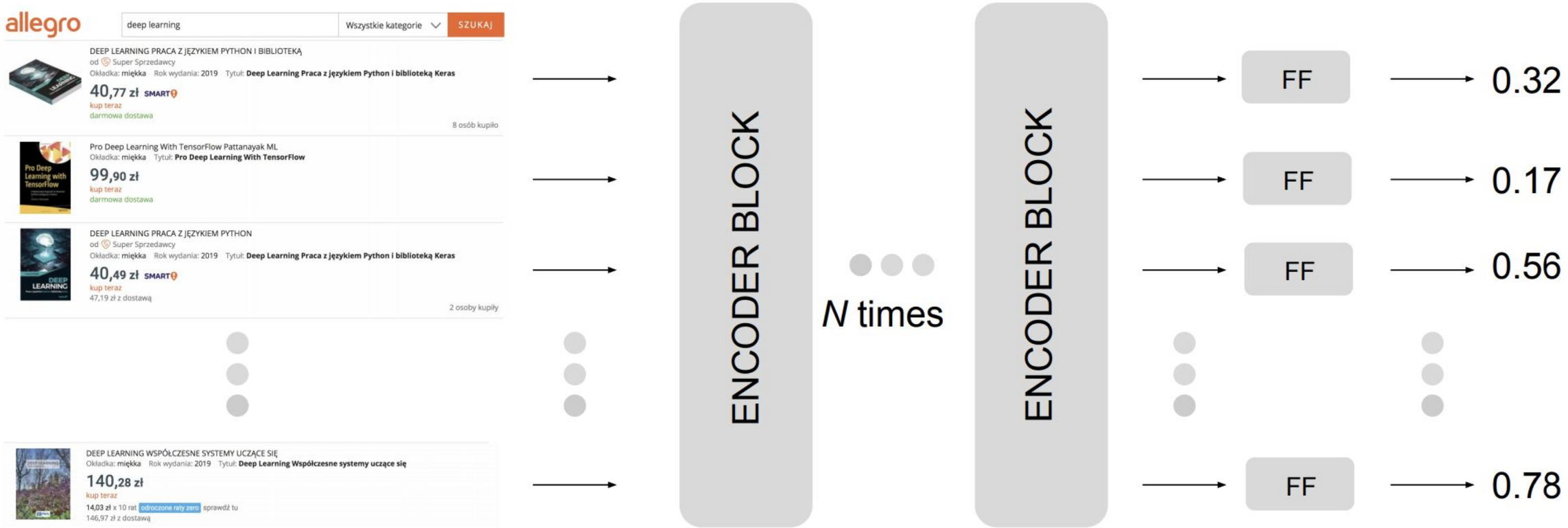
Encoder block

Transformer encoder blocks can be stacked on top of each other.



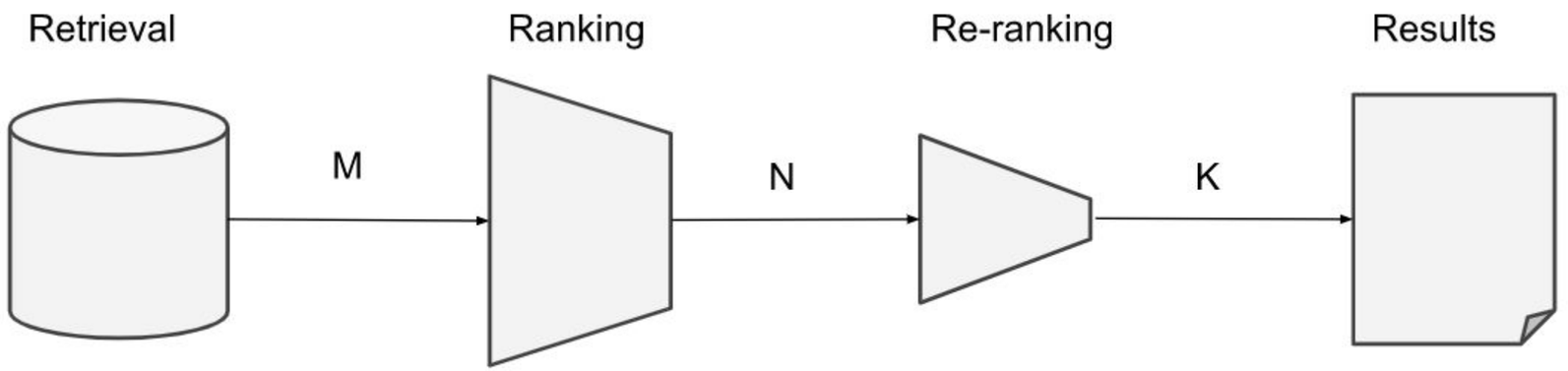
Our architecture

Architecture



Cascade ranking

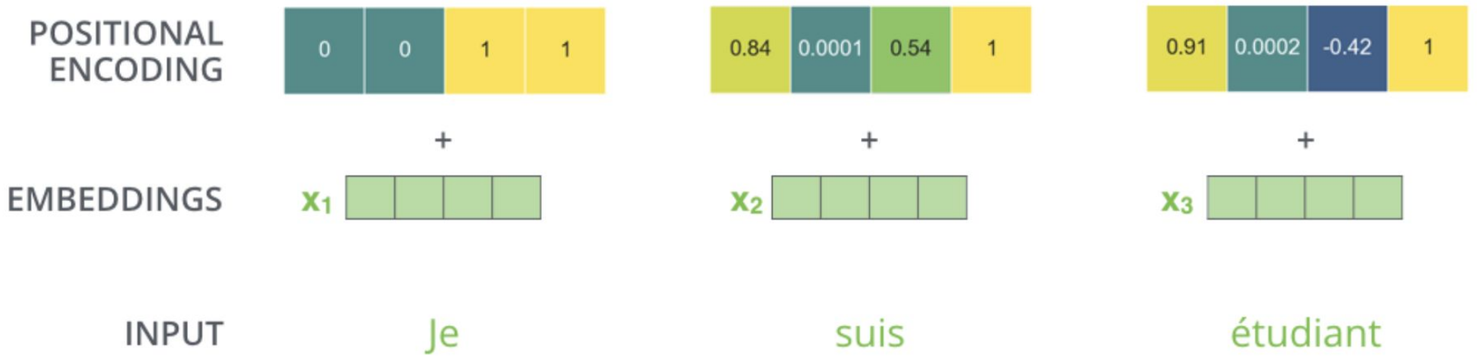
- There are multiple phases of ranking, each one is limiting the number of processed items
- This is a quality vs. efficiency tradeoff



M >> N >> K

Re-ranking by positional encoding

- Self-attention is invariant to the order of input items
- Positional Encoding was introduced together with Transformer to allow the model use position information



A real example of positional encoding with a toy embedding size of 4

Image: http://jalammar.github.io/illustrated-transformer/

Training regime - padding/sampling

- Lists of documents are of varying length
- Not all deep learning libraries support ragged tensors and so it requires fixed-size lists in batch
- We implemented padding/sampling for shorter/longer lists
- Loss function must account for padded items, as they should not contribute to the loss value

Experimental results

Datasets

WEB30K[1]

- Standard LTR evaluation dataset
- 30k queries
- Items' relevance graded by experts
- Dense, graded relevancies

Allegro's search logs

- In-house dataset
- 1M queries
- Implicit feedback (clicks)
- Sparse, binary relevancies

1. *Microsoft Learning to Rank Datasets*, https://www.microsoft.com/en-us/research/project/mslr/

WEB30K

- Every self-attentive model surpasses MLPs and the baseline XGBoost ranker
- Only one MLP is worse than the gradient boosting model

Loss	Self-attention	MLP
Ordinal loss	50.58	46.53
NDCGLoss2++	50.03	47.06
LambdaRank	49.59	46.57
ListNet	48.96	45.75
RMSE	49.01	46.25
RankNet	48.13	45.49
ListMLE	47.23	45.02
	XGBoost	
rank:pairwise	45.34	

Results reported as NDCG@5

Search logs

- Not every loss function applies to binary relevance cases
- MLPs no longer surpass XGBoost
- Weighted RankNet variants are versatile and effective!

Loss	Self-attention	MLP
NDCGLoss2++	3.00	1.51
LambdaRank	2.97	1.39
ListNet	2.93	1.24
RankNet	2.68	1.19
	XGBoost	
rank:pairwise[1]	1.83	

Results reported as NDCG@60 improvements (%) over the production ranker

1. rank:ndcg performs strictly worse, https://github.com/dmlc/xgboost/issues/4177

AllRank & Summary

AllRank

We provide an open-source implementation of our Transformer-based model and a rich set of loss functions:

https://github.com/allegro/allRank

The framework is based on PyTorch[1] and comes with a handy *getting started* example, kickstarting any LTR project, research or commercial (Apache License 2.0).

1. https://pytorch.org/

AllRank example

```
"model":
{
  "fc_model":
  {
    "sizes":
    [
      64,
      64,
      32
    ],
    "input_norm": true,
    "activation": null,
    "dropout": 0.1
  },
  "transformer":
  {
    "N": 2,
    "d_ff": 128,
    "h": 2,
    "positional_encoding": null,
    "dropout": 0.2
  },
  "post_model":
  {
    "output_activation": null,
    "d_output": 1
  }
},
```

```
"data":
{
  "path": "dataset_path",
  "validation_ds_role": "vali",
  "fold": "1",
  "num_workers": 1,
  "batch_size": 128,
  "listing_length": 20
},
"optimizer":
{
  "name": "Adam",
  "args":
  {
    "lr": 0.001
  }
},
"lr_scheduler":
{
  "name": "StepLR",
  "args":
  {
    "step_size": 1,
    "gamma": 0.9
  }
},
```

The only thing user needs to supply for LTR experiments is a JSON file containing training config and a dataset path (LIBSVM format).

It is also possible to use the AllRank building blocks in a custom PyTorch pipeline.

Future work

Our work has two sides - research and commercial. We will bring our models into production and simultaneously conduct series of experiments ranging from LTR theory to novel data approaches.

- Further AllRank development
- NDCG optimization by bound vs. direct approximation via approximate sorting operators
- Multi-modal ranking
- What about ranking diversity?

Thanks!

The paper covering the work presented here is pending review for one of the relevant conferences in the field. It was conducted by a Machine Learning Research team members:

- Przemysław Pobrotyn
- Tomasz Bartczak
- Mikołaj Synowiec
- Radosław Białobrzeski
- Jarosław Bojar