Transformer Contextualized Re-Ranking

Sebastian Hofstätter

sebastian.hofstaetter@tuwien.ac.at
/s_hofstaetter

Today

Transformer Contextualized Re-Ranking

1 Re-Ranking with BERT

- Concatenate query and document for a vanilla BERT use
- The mono-duo pattern; long document handling

2 Efficient Transformer-based Models

- Splitting BERT: PreTTR, ColBERT
- Our Transformer-Kernel family
- IDCM: A hybrid model

Now We Reached the State-of-the-art

- Finally!
- Fast moving field:
  - Most of the contents of this lecture did not exist in the beginning of 2018
  - And by July we probably have a new SOTA technique
- Many questions are open
  - We try to answer a few here

- General direction: More computation = better results,
possible by better hardware & more data

Context of this Lecture

- We are studying ad-hoc re-ranking models
  - Many other applications of Transformers in IR:
    - Document & query expansion
    - QA pipelines
    - Conversational search, using the query history
    - Dense retrieval
    - Knowledge graph-based search
- We are focusing on the efficiency-effectiveness tradeoff
  - How fast or slow is a model compared to the result quality
  - Many other aspects can be studied, f.e.: social biases, domain adaption, or multilingual models (and many more)

Web Search with BERT

Google

The Keyword   Latest Stories   Product Updates   Company News

SEARCH

Understanding searches better than ever before

Pandu Nayak
Google Fellow and Vice President, Search

Published Oct 25, 2019

If there's one thing I've learned over the 15 years working on Google Search, it's that people's curiosity is endless. We see billions of searches every day, and 15 percent of those queries are ones we haven't seen before--so we've built ways to return results for queries we can't anticipate.

When people like you or I come to Search, we aren't always quite sure about the best way to formulate a query. We might not know the right words to use, or how to spell something, because often times, we come to Search looking to learn--we don't necessarily have the knowledge to begin with.

At its core, Search is about understanding language. It's our job to figure out what you're searching for and surface helpful information from the web, no matter how you spell or combine the words in your query. While we've continued to improve our language understanding capabilities over the years, we

Google (October 2019)

Microsoft Azure   Contact Sales: 0800-206-208   Search

Overview   Solutions   Products   Documentation   Pricing   Training   Marketplace   Partners   Support   Blog

Blog / Virtual Machines

Bing delivers its largest improvement in search experience using Azure GPUs

Posted on November 18, 2019

Jeffrey Zhu, Program Manager, Bing Platform

Over the last couple of years, deep learning has become widely adopted across the Bing search stack and powers a vast number of our intelligent features. We use natural language models to improve our core search algorithm's understanding of a user's search intent and the related webpages so that Bing can deliver the most relevant search results to our users. We rely on deep learning computer vision techniques to enhance the discoverability of billions of images even if they don't have accompanying text descriptions or summary metadata. We leverage machine-based reading comprehension models to retrieve captions within larger text bodies that directly answer the specific questions users have. All these enhancements lead toward more relevant, contextual results for web search queries.

Recently, there was a breakthrough in natural language understanding with a type of model called transformers (as popularized by Bidirectional Encoder Representations from Transformers, BERT). Unlike previous deep neural network (DNN) architectures that processed words individually in order, transformers understand the context and relationship between each word and all the words around it in a sentence. Starting from April of this year, we used large transformer models to deliver the largest quality improvements to our Bing customers in the past year. For example, in the query "what can aggravate a concussion", the word "aggravate" indicates the user wants to learn about actions to be taken after a concussion and not about causes or symptoms. Our search

Microsoft (November 2019)

Re-Ranking with BERT

Drastically improving search result quality

Recall: BERT - Workflow

- Someone with lots of compute or time pre-trains a large model
  - BERT uses Masked Language Modelling [MASK] and Next Sentence Prediction [CLS]
- We download it and fine-tune on our task

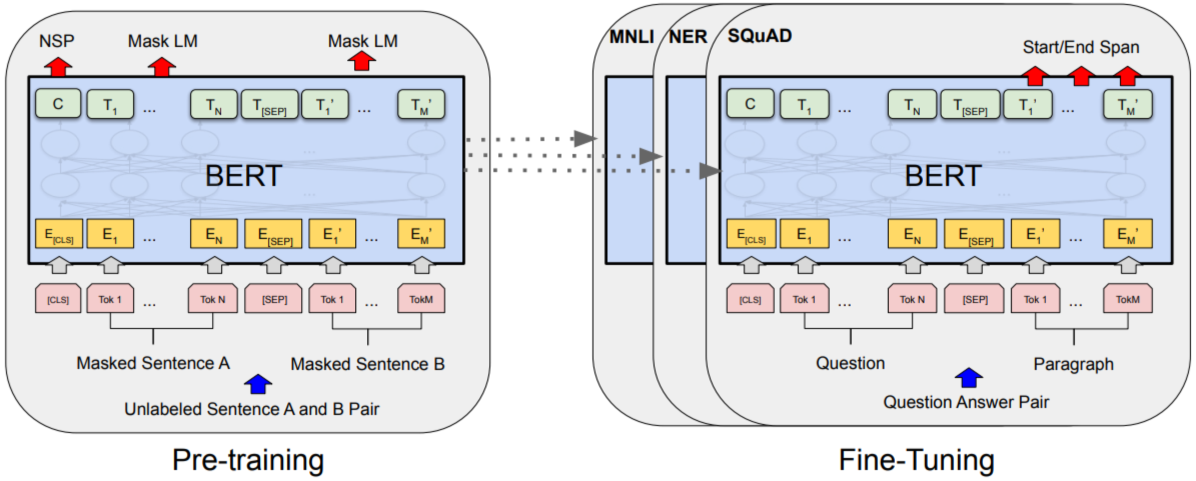
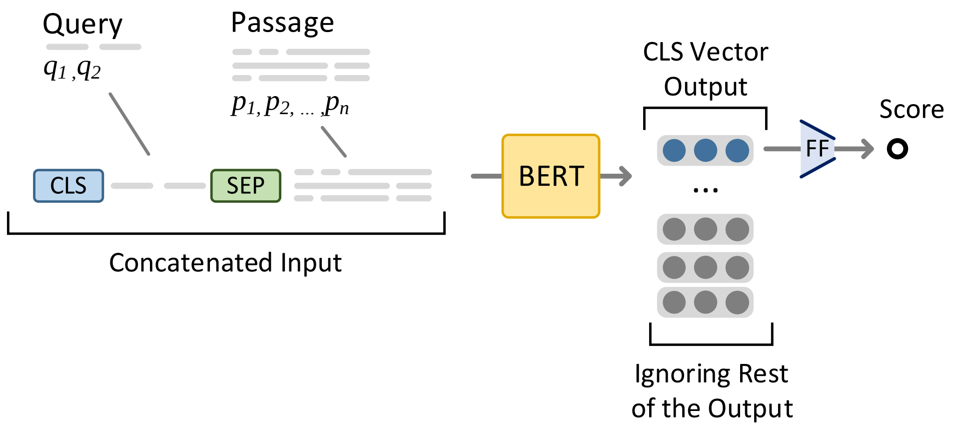


Figure taken from the BERT paper

BERT Re-Ranking: BERT_{CAT}



- Also know as monoBERT, vanilla BERT re-ranking, or simply BERT
- Concatenating the two sequences to fit BERT's workflow
  - [CLS] query [SEP] passage
  - Pool [CLS] token
  - Predict the score with a single linear layer
- Needs to be repeated for every passage

Passage Re-ranking with BERT. Rodrigo Nogueira, Kyunghyun Cho. 2019
https://arxiv.org/abs/1901.04085

BERT_{CAT}

- Simple formula (as long as we abstract BERT):

BERT

$$r = \text{BERT}(\underbrace{[CLS]; q_{1..n}; [SEP]; p_{1..m}}_{\text{Concatenation}})_{CLS}$$

Scoring

$$s = r * \underbrace{W}_{\text{Starts uninitialized}}$$

- We still have the choice of BERT-model

$q_{1..n}$	Query tokens
$p_{1..m}$	Passage tokens
BERT	Pre-trained BERT model
[CLS] [SEP]	Special tokens
x_{CLS}	Pool the CLS vector
W	Linear Layer (from 768 dims to 1)
s	Output score

The Impact of BERT$_{CAT}$

- This model (first shown by Nogueira and Cho) jumpstarted the current waive of neural IR 🎉
- Works awesome out of the box
  - Concatenating the two sequences to fit BERT’s workflow
  - As long as you have time or enough compute it trains easily
- Major jumps in effectiveness across collections and domains
  - But, of course, comes at the cost of performance and virtually no interpretability
  - Larger BERT models roughly translate to slight effectiveness gains at high efficiency cost
    - The problem is we need to repeat the inference by the re-ranking depth!

So how good is BERT_{CAT} ?

- MSMARCO-Passage
  - MRR@10 from .194 (BM25) to .385 (ALBERT-Large)
  - BERT basically doubles the result quality

- MSMARCO-Document
  - MRR@10 from .252 (BM25) to .384 (DistilBERT with 2K tokens)

- Similar results for TREC-DL ’19, ’20, TripClick
  - Large Training Data Settings

See also the MSMARCO leaderboard: https://microsoft.github.io/msmarco/
Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation; Hofstätter et al. https://arxiv.org/abs/2010.02666

The Mono-Duo Pattern

- Mono-Duo is a multi-stage process:
  - Mono = score(q, p) – f.e. Top-1000
  - Duo = score(q, p1, p2) – f.e. Top-50 (but needs to be done 50^2 times)

- Improves on the single “mono” stage
  - Good for leaderboard settings, and showing the maximum achievable performance
- Can use BERT or T5 as base model
  - T5 is also a Transformer based language model, even larger but of course slower

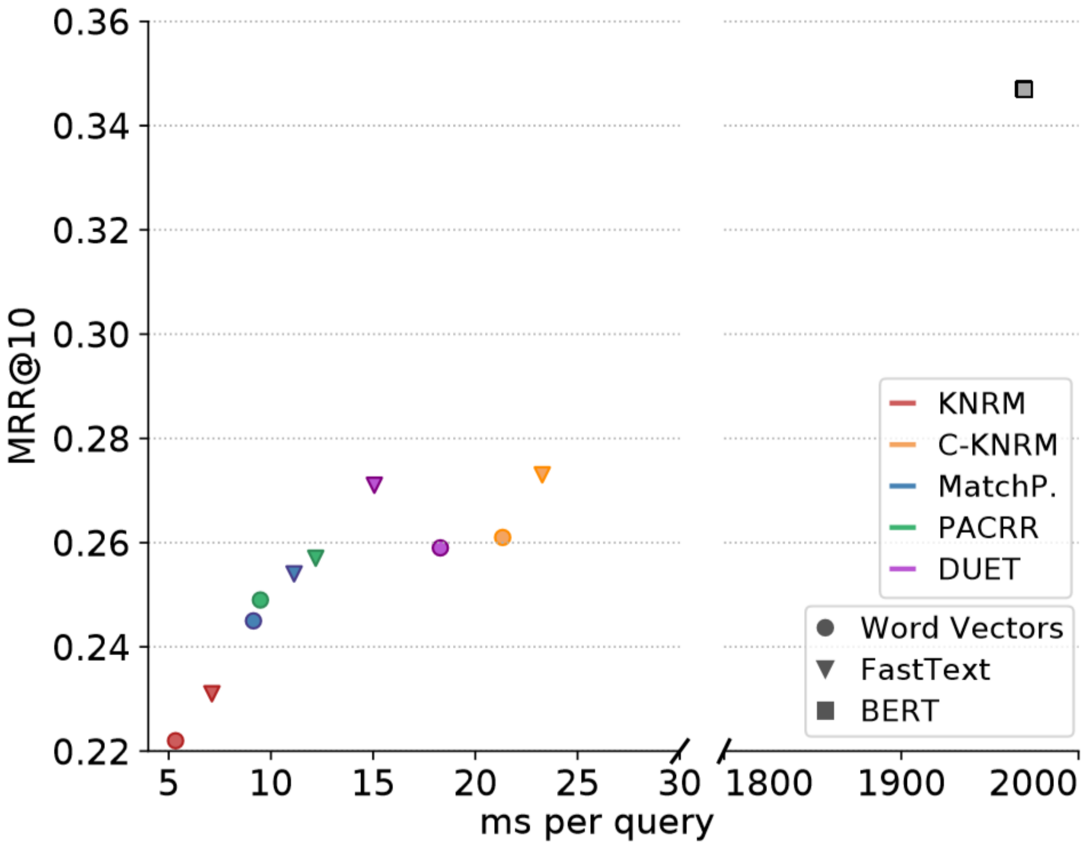
The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models
Ronak Pradeep, Rodrigo Nogueira, Jimmy Lin https://arxiv.org/abs/2101.05667

BERT_{CAT} for Longer Documents

- BERT is capped at max. 512 input tokens (query + document)
- Simplest solution: just take cap the document at 512-query length
  - Works surprisingly well already (for MSMARCO-Documents)
  - But might not work well in other domains, where documents are really long, or contain a variety of topics at different depths
- Still simple, but working on full documents: Sliding window over the document -> take max window score as document score
  - Now, we can also make smaller sliding windows
    - Might be useful to use in the UI -> highlight the most relevant passage as snippet

Deeper Text Understanding for IR with Contextual Neural Language Modeling. Zhuyun Dai, Jamie Callan
https://arxiv.org/abs/1905.09217

BERT$_{CAT}$ In-Efficiency



- Evaluated on 250 docs / query on short MSMARCO-Passage *(max 200 tokens)*
- Basic IR-specific networks are fast, but moderately effective
- Transformer-based BERT is very effective, but very slow
  - + Infrastructure cost (blocking 1 GPU for 2 seconds at a time)

Sebastian Hofstätter and Allan Hanbury. 2019.
Let's measure runtime! Extending the IR replicability infrastructure to include performance aspects . In OSIRRC @ SIGIR.

Efficient Transformer-based Models

Adapting BERT or starting from scratch altogether

Achieving Efficiency

- Multiple paths to reduce query latency
  - Query latency is our focus today, but full lifecycle efficiency also a concern
  - Lifecycle efficiency includes training, indexing and retrieval steps

❶ Reduce model size

- Smaller models run faster, duh!
- Only possible until a certain threshold after which quality reduces drastically

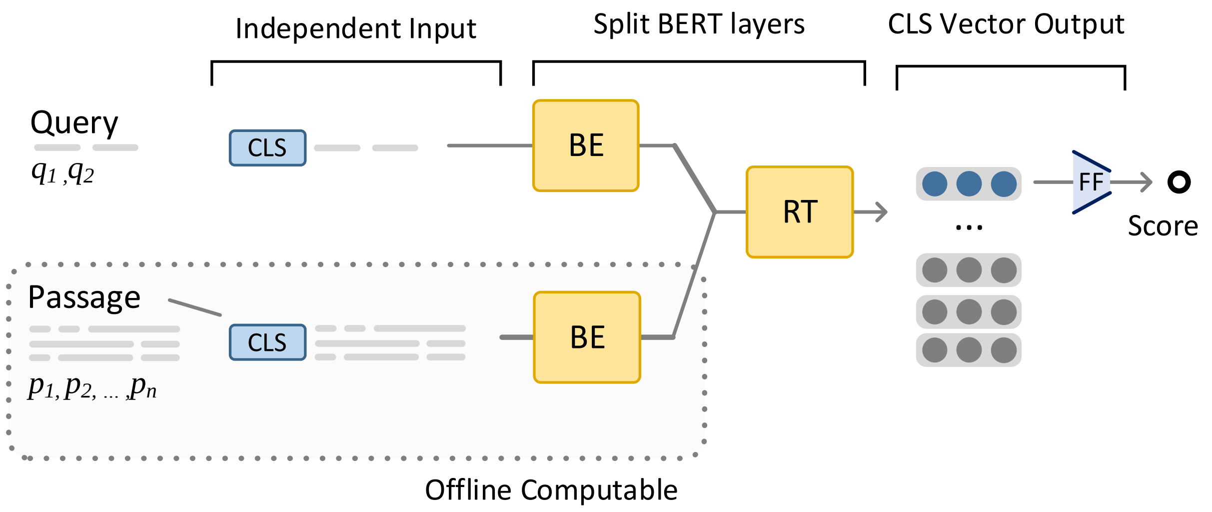
❷ Move computation away from query-time

- Pre-compute passage representation, so they become a simple lookup
- Lightweight aggregation, that can be done at query time

Splitting BERT for Efficiency

- Concatenated BERT_{CAT} needs x re-ranking depth evaluations at query time
  - Bad for query latency, deep re-ranking implausible
- Most of that computation is spent on passage encoding
  - If we can move the passage encoding to the indexing phase
  - Encode each passage exactly 1 time
  - At query time only need to encode 1 query (with few words, so it's fast)
- Multiple approaches have been proposed to then *glue* the query and passage representations back together
  - With only a small reduction in effectiveness

Splitting BERT: PreTTR



- PreTTR splits BERT layers:
  - The first *n* layers are separated
  - The following layers are concatenated again
  - *n* is a hyperparameter
- We can pre-compute the first-n layers of the passages
- Pro: ≈ quality as BERT_{CAT}
- Neg: Still low query latency + storage requirements

Efficient Document Re-Ranking for Transformers by Precomputing Term Representations; MacAvaney et al. SIGIR 2020 https://arxiv.org/abs/2004.14255

PreTTR

- Split in two parts:

Part 1

$$\hat{q}_{1..n} = \mathrm{BERT}_{1..b}([CLS]; q_{1..n})$$

$$\hat{p}_{1..m} = \mathrm{BERT}_{1..b}([CLS]; p_{1..m})$$

Independent computation

Part 2

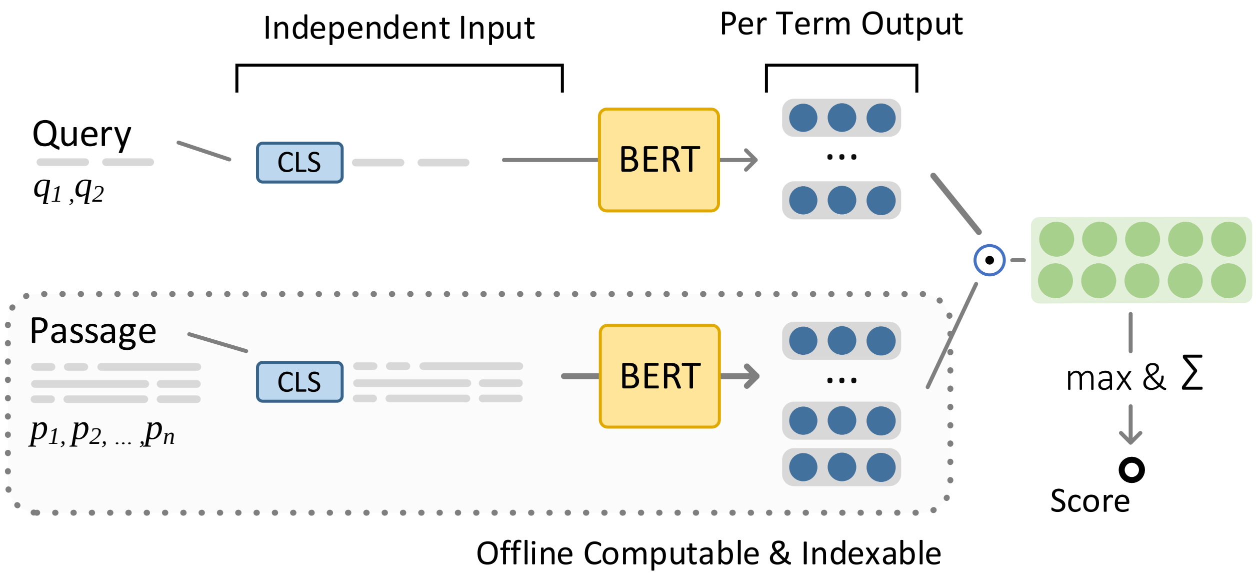
$$s = \underline{W} * \mathrm{BERT}_{b..l}(\hat{q}_{1..n}[SEP]; \hat{p}_{1..m})$$

Starts uninitialized

- Optional compression of $\hat{q}_{1..n}$ & $\hat{p}_{1..m}$

$q_{1..n}$	Query tokens
$p_{1..m}$	Passage tokens
BERT	Pre-trained BERT model
[CLS] [SEP]	Special tokens
W	Linear Layer (from 768 dims to 1)
s	Output score

Splitting BERT: ColBERT



- ColBERT creates a match-matrix of BERT term-representations
- Then uses a simple max-pooling for the doc-dim & sum for the query dim
- Pro: Very fast query latency
- Con: Need to save all passage term vectors (huge storage cost)

ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT; Khattab et al. SIGIR 2020 https://arxiv.org/abs/2004.12832

ColBERT

- Simple formula (as long as we abstract BERT):

Encoding

$$\hat{q}_{1..n} = \mathrm{BERT}([CLS]; q_{1..n})$$

$$\hat{p}_{1..m} = \mathrm{BERT}([CLS]; p_{1..m})$$

Can be done at indexing time

Aggregation

$$s = \sum_{i=1..n} \max_{t=1...m} \hat{q}_i \cdot \hat{p}_t$$

Quite efficient at query time

- Optional compression of $\hat{q}$, $\hat{p}$ with a single linear layer

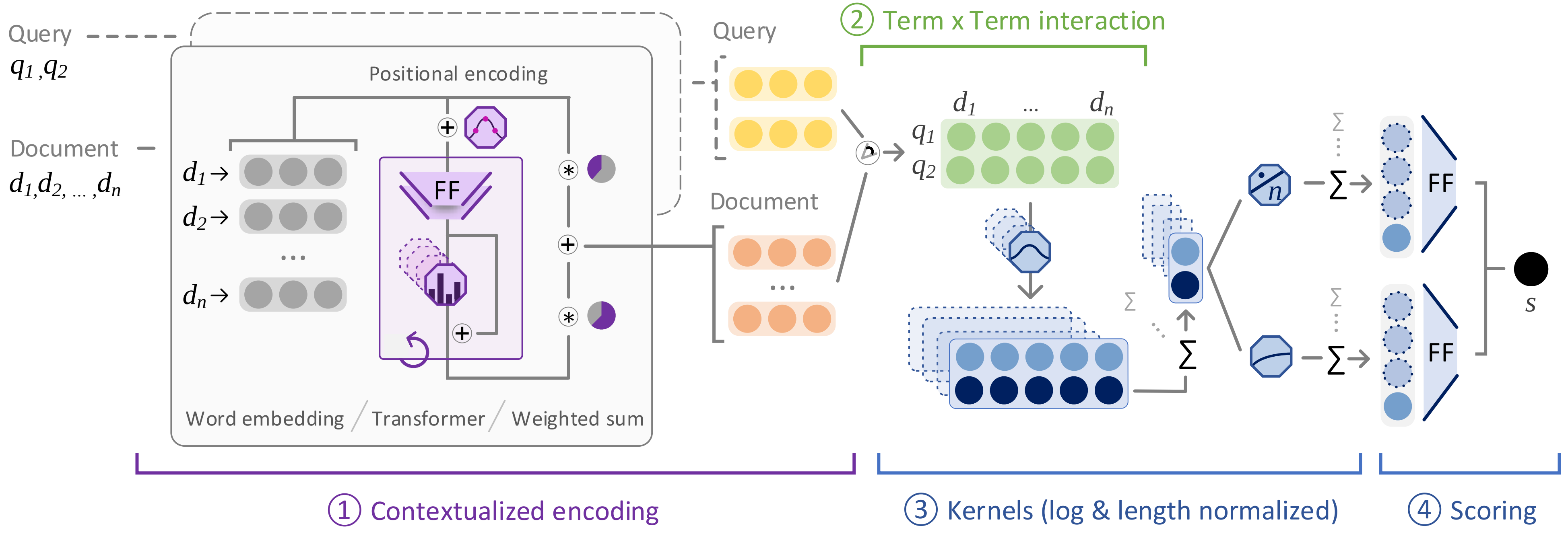
$q_{1..n}$	Query tokens
$p_{1..m}$	Passage tokens
BERT	Pre-trained BERT model
[CLS]	Special tokens
s	Output score

TK: Transformer-Kernel Ranking

- Desired properties: Lightweight, interpretable, effective
- We proposed the TK model
  - Combine Transformer-contextualization with kernel-pooling
  - Strong results compared to IR-specific models
  - State-of-the-art model for time-budget constrained environments
- Use Transformer-blocks as contextualization layer
  - Create hybrid contextualization by merging context & non-context
- Limit the number of Transformer layers, as each additional layer takes considerable amount of time with diminishing returns

Sebastian Hofstätter, Markus Zlabinger and Allan Hanbury. 2020.
Interpretable & time-budget-constrained contextualization for re-ranking. In ECAI.

TK: Transformer-Kernel Ranking



TK

Encoding

$$\hat{q}_{1..n} = TF(q_{1..n}) \qquad \hat{d}_{1..m} = TF(d_{1..m})$$

Matching

$$M_{ij} = \mathbf{cos}(\hat{q}_i, \hat{d}_j)$$

A single match matrix

Kernels

$$K_k(M) = \sum_{i=1}^{n} \log\left(\sum_j \exp\left(-\frac{\left(M_{ij} - \mu_k\right)^2}{2\sigma_k^2}\right)\right)$$

Same as KNRM

Scoring

$$s = \boldsymbol{FC}(K) = W * K + b$$

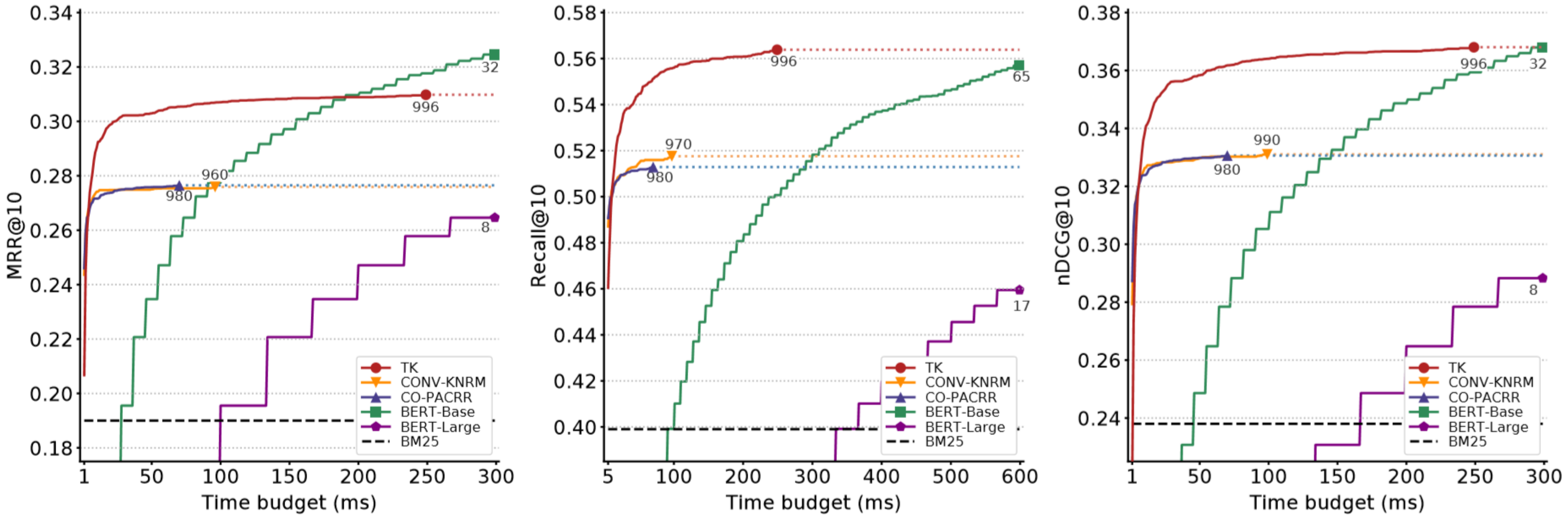
$q_{1..n}$	Query token vecs
$d_{1..m}$	Doc. token vecs
M	Match-matrix
TF	Transformer
K	All Kernels
K_k	k-th kernel
μ_k	Similarity level
σ_k	Kernel-width/range
W, b	Weights & biases
s	Output score

Designing for a Time-Budget

- Large influence on how we employ a model
- Main outside factor: how many documents to re-rank
- Faster models can re-rank more documents in the same time as slower ones
- Evaluate based on a time budget
  - Allows us to simultaneously evaluate effectiveness & efficiency in a realistic setting

TK on a Time Budget

- More documents in the same time = better results



(a) MSMARCO-Passage

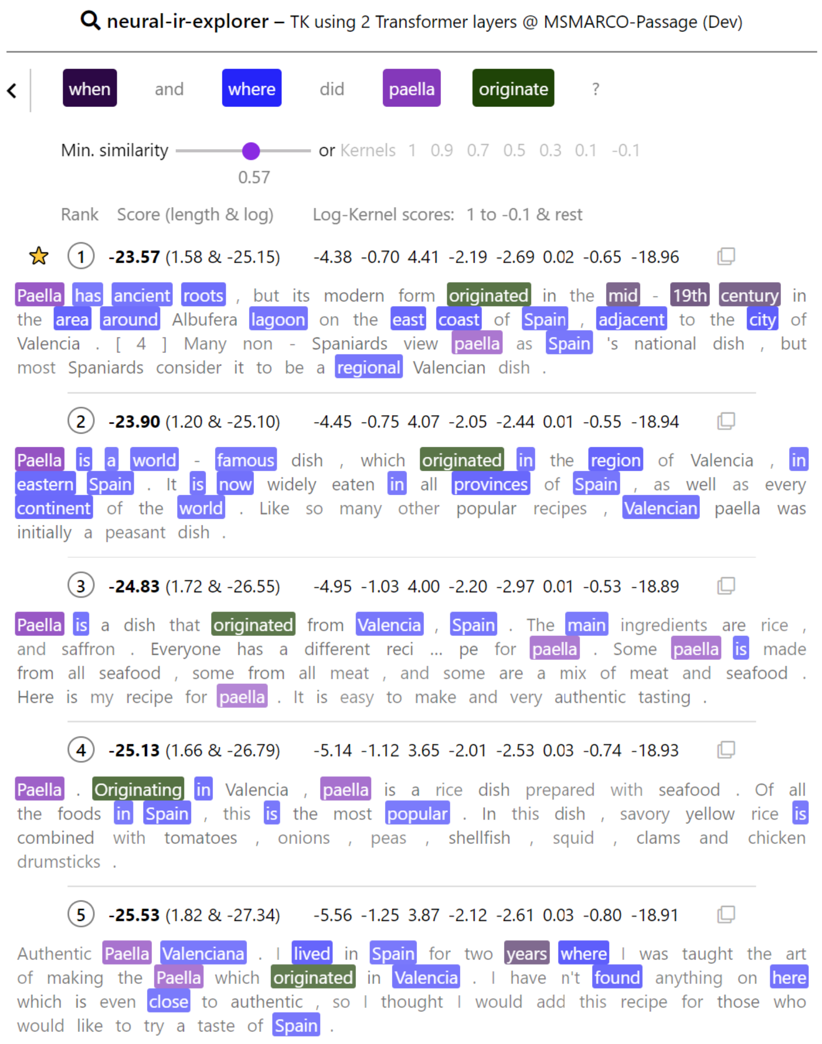
Comparing the Models

- Many dimension to compare possible:
  - Quality (Effectiveness, diff. measures)
  - Query latency, Storage requirement, Capabilities, etc…

Model	Effect.	Query Latency	GPU Memory	Query-Passage Interaction	Passage Cache	NN Index	Storage (× Dim.)
$\mathbf{BERT_{CAT}}$	1	950 ms	10.4 GB	All TF layers	–	–	–
$\mathbf{BERT_{DOT}}$	× 0.87	23 ms	3.6 GB	Single dot product	✓	✓	$\|P\|$
ColBERT	× 0.97	28 ms	3.4 GB	$m * n$ dot products	✓	✓	$\|T\|$
PreTT	× 0.97	455 ms	10.9 GB	Min. 1 TF layer (here 3)	✓	–	$\|T\|$
TK	× 0.89	14 ms	1.8 GB	$m * n$ dot pr. + Kernel-pool	✓	–	$\|T\|$

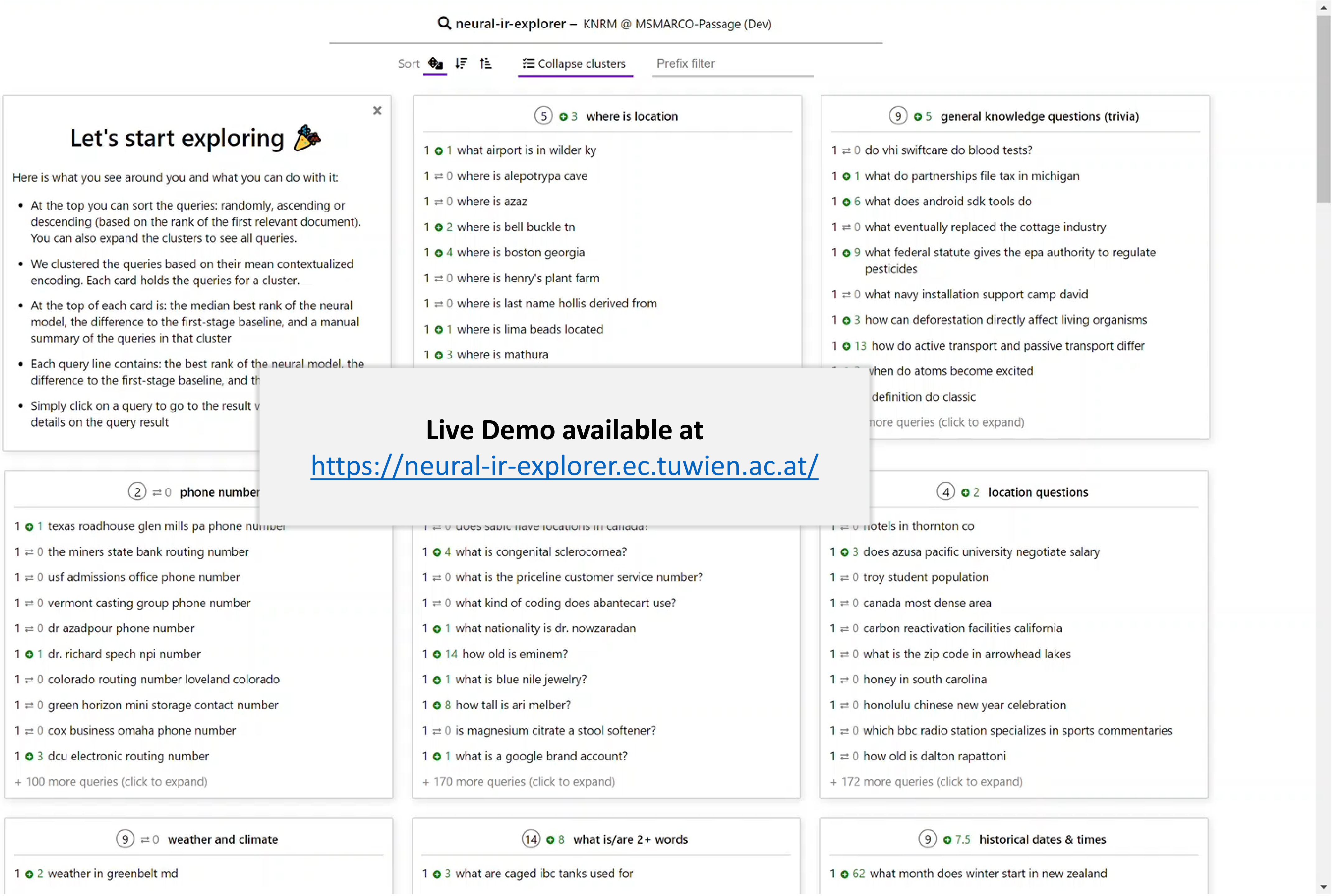
Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation; Hofstätter et al. https://arxiv.org/abs/2010.02666

Understanding TK



- Demo application to showcase TK
  - Displays internal similarity & kernel results
- Users can browse around the results
  - Get an overview over the queries
  - Dig deep into a single result
- Complements metric-based evaluation
- Allows users to develop a “feeling” for the test collection & model used

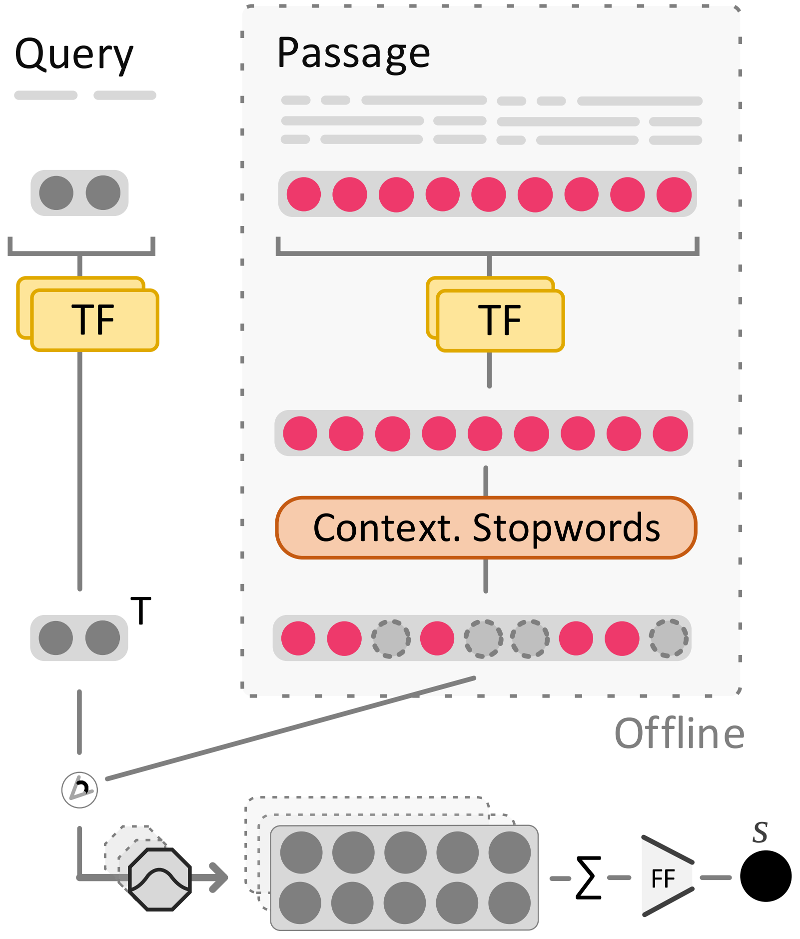
Sebastian Hofstätter, Markus Zlabinger and Allan Hanbury. 2020. Neural-IR-Explorer: A Content-Focused Tool to Explore Neural Re-ranking Results. In ECIR.



Live Demo available at

https://neural-ir-explorer.ec.tuwien.ac.at/

TK-Sparse with Contextualized Stopwords



- The TK model contextualizes query & passage independent
- We offload passage computation
- Stopwords are removed by a trained sparsity module after contextualization
- Sparsity is trained end-to-end with L1 norm augmented loss function

Learning to Re-Rank with Contextualized Stopwords
Hofstätter et al. CIKM 2020

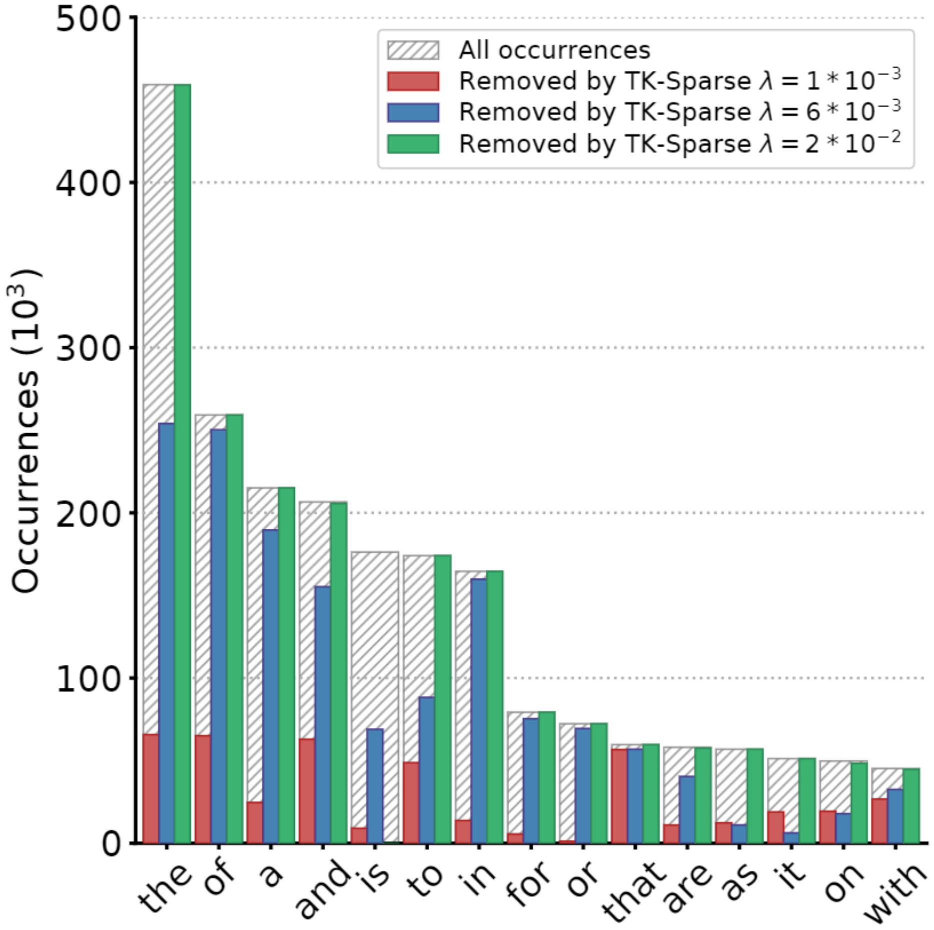
Effectiveness Results (MSMARCO-Passage)

Sig.	Model	Stop	DEV (limited judgments) nDCG@10	DEV MRR@10	DEV R@10	TREC-2019 (dense judgments) nDCG@3	TREC-2019 nDCG@10	TREC-2019 MRR@10	TREC-2019 R@10	TREC-2019 MAP@1K
a	***TK***	-	^{bcdj}0.369	^{bcdj}0.311	^{bcdj}0.564	0.655	0.649	0.821	**0.242**	0.396
b	***TK /w LuceneStop***	24 %	^{cdj}0.359	^{cdj}0.301	^{cdj}0.555	0.661	0.630	^{dg}0.795	0.220	0.392
c	***TK /w CollectionTop25***	35 %	^{cdj}0.353	^{dj}0.296	^{j}0.547	0.649	0.624	0.756	0.214	^{d}0.388
d	***TK /w CollectionTop50***	41 %	0.350	0.293	0.543	0.635	0.627	0.745	0.230	0.376
e	***TK-Sparse*** $\lambda = 8 * 10^{-4}$	3 %	^{abcdghij}**0.373**	^{abcdhij}**0.314**	^{abcdj}0.569	^{d}0.669	0.638	0.789	^{c}0.230	0.384
f	***TK-Sparse*** $\lambda = 1 * 10^{-3}$	19 %	^{abcdhj}**0.373**	^{abcdhj}**0.314**	^{abcdhj}**0.570**	^{cd}0.691	^{bcdeg}**0.658**	^{dceg}**0.840**	^{c}0.232	^{bcdegh}**0.400**
g	***TK-Sparse*** $\lambda = 3 * 10^{-3}$	12 %	^{abcdj}0.372	^{abcdj}**0.314**	^{abcdj}0.568	^{d}0.665	0.632	0.758	0.220	^{d}0.382
h	***TK-Sparse*** $\lambda = 6 * 10^{-3}$	26 %	^{abcdj}0.371	^{bcdj}0.312	^{bcdj}0.567	0.681	0.653	0.821	0.231	0.391
i	***TK-Sparse*** $\lambda = 9 * 10^{-3}$	18 %	^{abcdhj}0.371	^{abcdj}0.312	^{bcdj}0.567	^{bcdeg}**0.714**	^{bcdeg}0.657	^{dcg}0.827	0.227	^{bdegh}**0.400**
j	***TK-Sparse*** $\lambda = 2 * 10^{-2}$	43 %	0.350	0.293	0.542	^{bcdgh}0.705	^{bcdeg}0.657	^{dcg}0.832	^{c}0.239	^{d}0.395

Effectiveness Results (MSMARCO-Passage)

Sig.	Model	Stop	DEV (limited judgments)			TREC-2019 (dense judgments)				
			nDCG@10	MRR@10	R@10	nDCG@3	nDCG@10	MRR@10	R@10	MAP@1K
a	***TK***	-	^{bcdj}0.369	^{bcdj}0.311	^{bcdj}0.564	0.655	0.649	0.821	**0.242**	0.396
b	***TK /w LuceneStop***	24 %	^{cdj}0.359	^{cdj}0.301	^{cdj}0.555	0.661	0.630	^{dg}0.795	0.220	0.392
c	***TK /w CollectionTop25***	35 %	^{cdj}0.353	^{dj}0.296	^{j}0.547	0.649	0.624	0.756	0.214	^{d}0.388
d	***TK /w CollectionTop50***	41 %	0.350	0.293	0.543	0.635	0.627	0.745	0.230	0.376
e	***TK-Sparse*** $\lambda = 8 * 10^{-4}$	3 %	^{abcdghij}**0.373**	^{abcdhij}**0.314**	^{abcdj}0.569	^{d}0.669	0.638	0.789	^{c}0.230	0.384
f	***TK-Sparse*** $\lambda = 1 * 10^{-3}$	19 %	^{abcdhj}**0.373**	^{abcdhj}**0.314**	^{abcdhj}**0.570**	^{cd}0.691	^{bcdeg}**0.658**	^{dceg}**0.840**	^{c}0.232	^{bcdegh}**0.400**
g	***TK-Sparse*** $\lambda = 3 * 10^{-3}$	12 %	^{abcdj}0.372	^{abcdj}**0.314**	^{abcdj}0.568	^{d}0.665	0.632	0.758	0.220	^{d}0.382
h	***TK-Sparse*** $\lambda = 6 * 10^{-3}$	26 %	^{abcdj}0.371	^{bcdj}0.312	^{bcdj}0.567	0.681	0.653	0.821	0.231	0.391
i	***TK-Sparse*** $\lambda = 9 * 10^{-3}$	18 %	^{abcdhj}0.371	^{abcdj}0.312	^{bcdj}0.567	^{bcdeg}**0.714**	^{bcdeg}0.657	^{dcg}0.827	0.227	^{bdegh}**0.400**
j	***TK-Sparse*** $\lambda = 2 * 10^{-2}$	43 %	0.350	0.293	0.542	^{bcdgh}0.705	^{bcdeg}0.657	^{dcg}0.832	^{c}0.239	^{d}0.395

Stopword Analysis



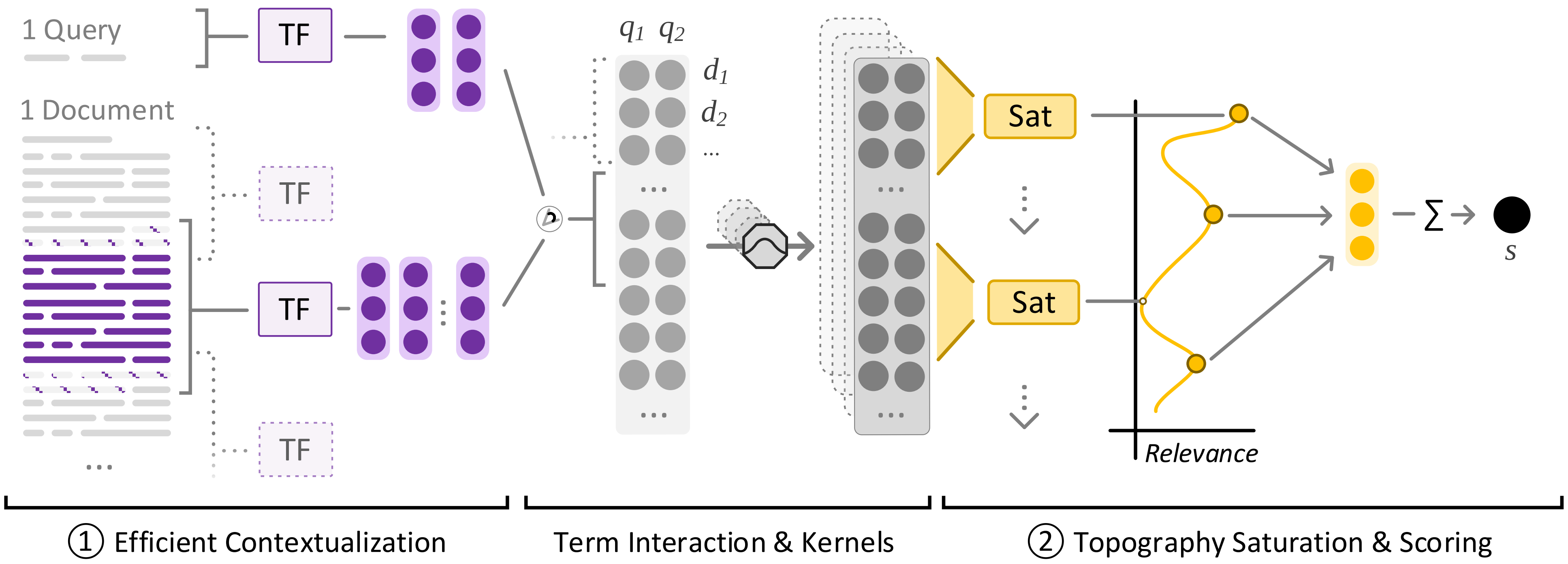
- Common stopword lists remove every occurrence of a term
  - Contextualized stopwords decide per occurrence
- Here we compare the most common stopwords from Lucene with different configurations of TK-Sparse
  - We observe an overlap of removed terms and substantial differences

TKL: Transformer-Kernel for Long Documents

- Long documents (>200 tokens) are very slow
  - 300-dimensional vector per each word
  - Re-rank 100s of documents per query
  - Padding techniques are insufficient
- State-of-the-art models don’t work
  - Do not contain a notion of region imporatnce
  - Current best approach split a document and score sentences/paragraphs individually
- We proposed an extension to TK for Long documents (TKL)

Sebastian Hofstätter, Hamed Zamani, Bhaskar Mitra, Nick Craswell and Allan Hanbury. 2020.
Local Self-Attention over Long Text for Efficient Document Retrieval. In SIGIR.

TKL: Transformer-Kernel for Long Documents



TKL: Why long documents?

- We found that longer document input gives us better results
- But only if we do top region detection in TKL
- Main idea behind exercise 1:
  - Find out if the model was correct in this assumption
  - Can we proof that we need longer input

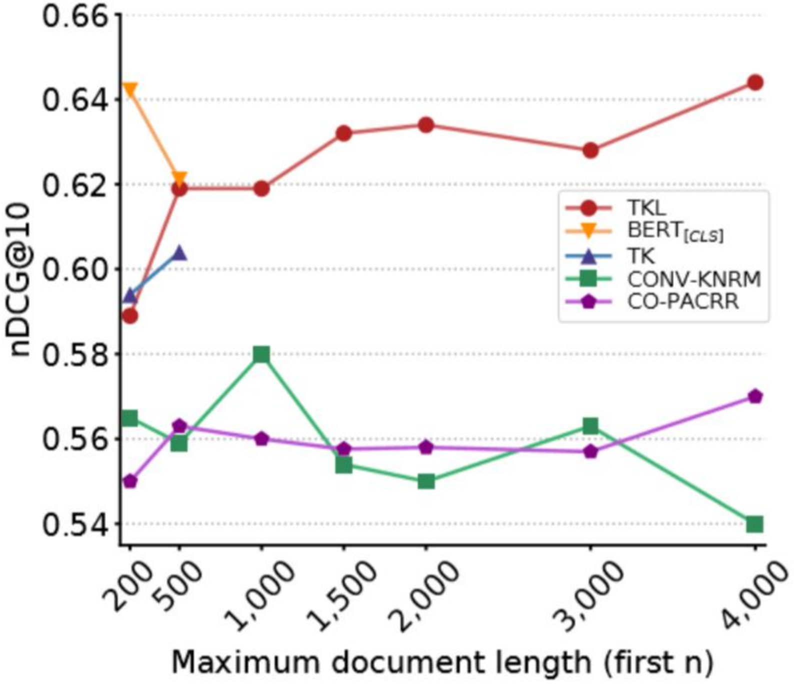
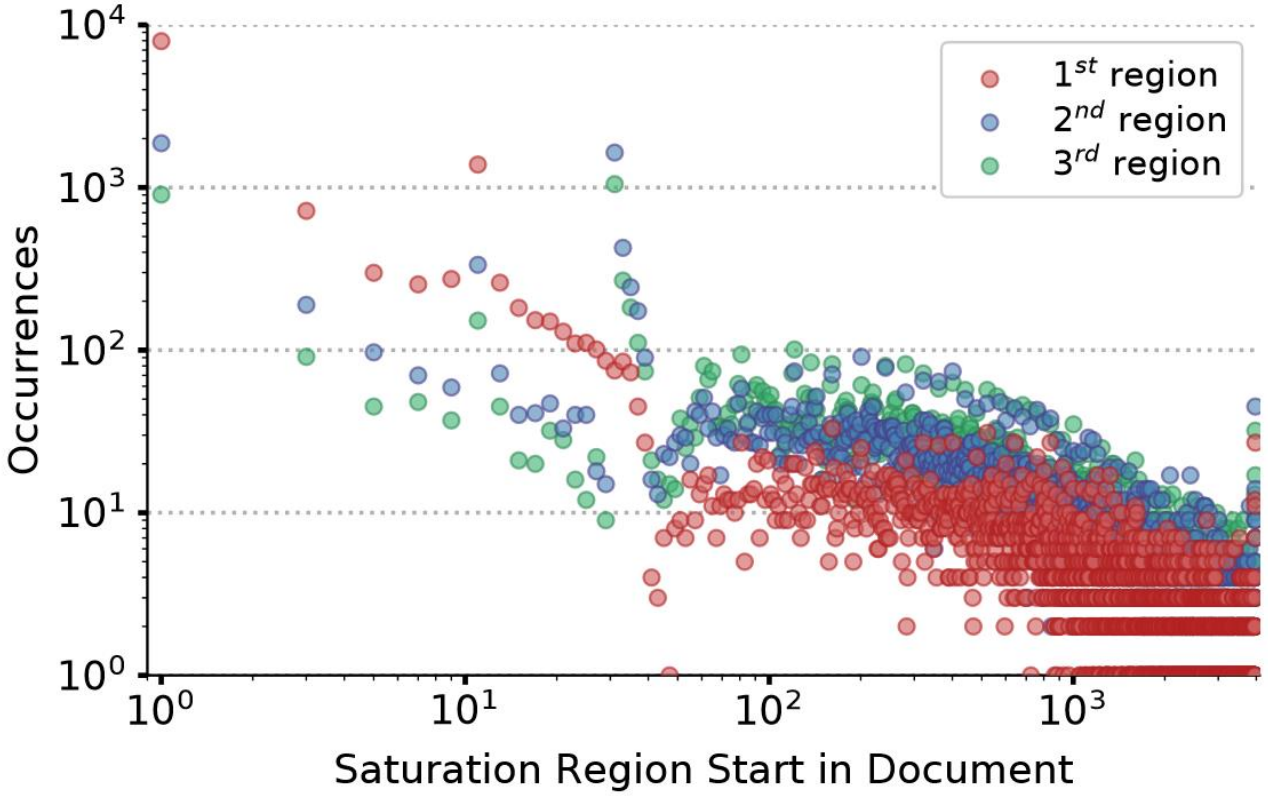


Figure 2: TREC-2019 results based on the document length.

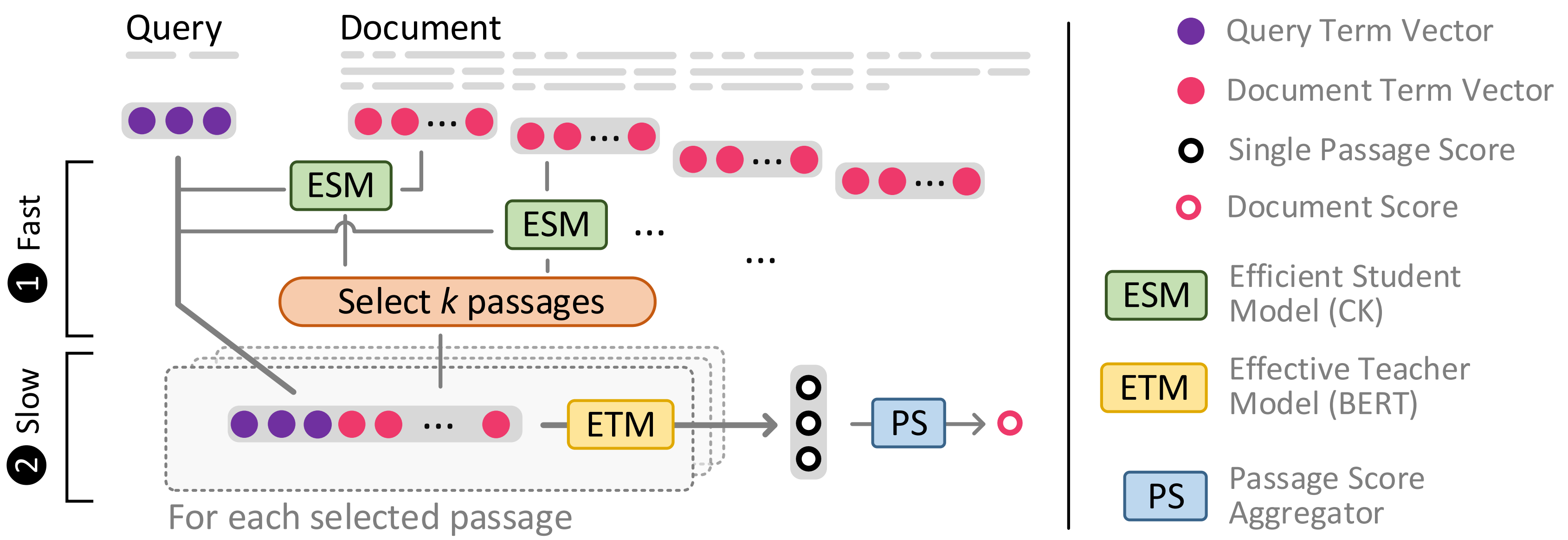
TKL: Where is the relevance?

- TKL provides relevant regions location in the document
- Occurrences follow a Zipfian-Distribution
  - In the TREC-DL 2019 Document collection
- Could be used as snippet generation
  - Better user interfaces



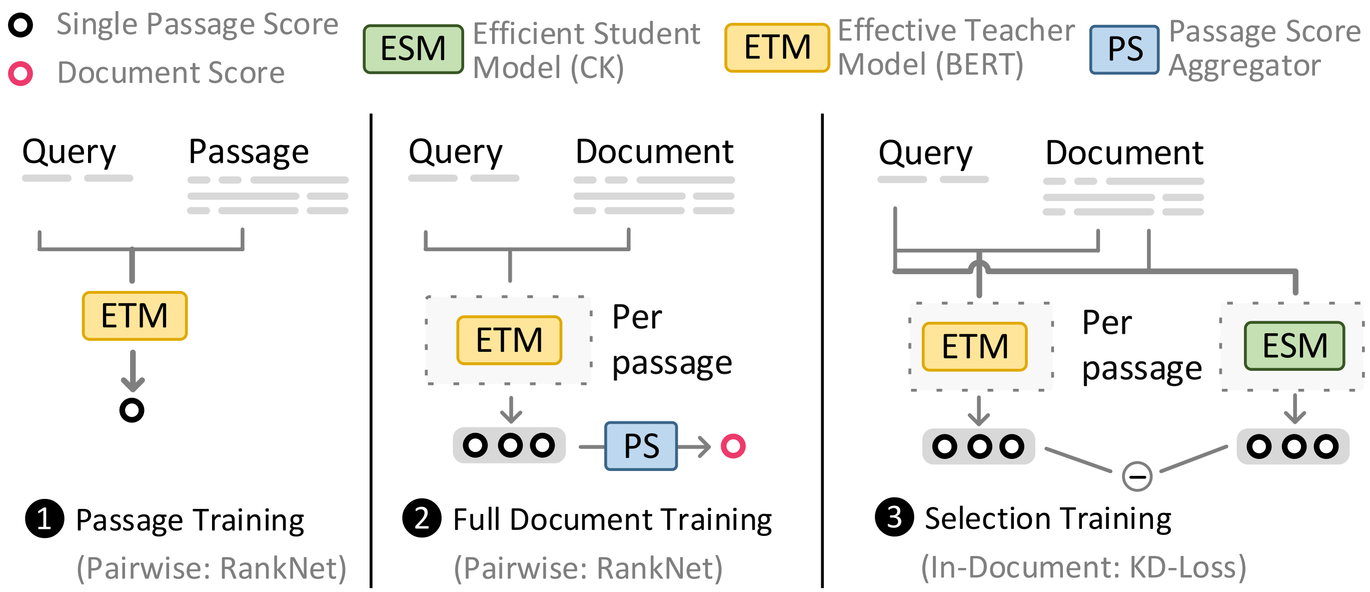
IDCM: A Hybrid Approach

- Combining a slow and fast module in one model to provide an intra-document cascade (IDCM)



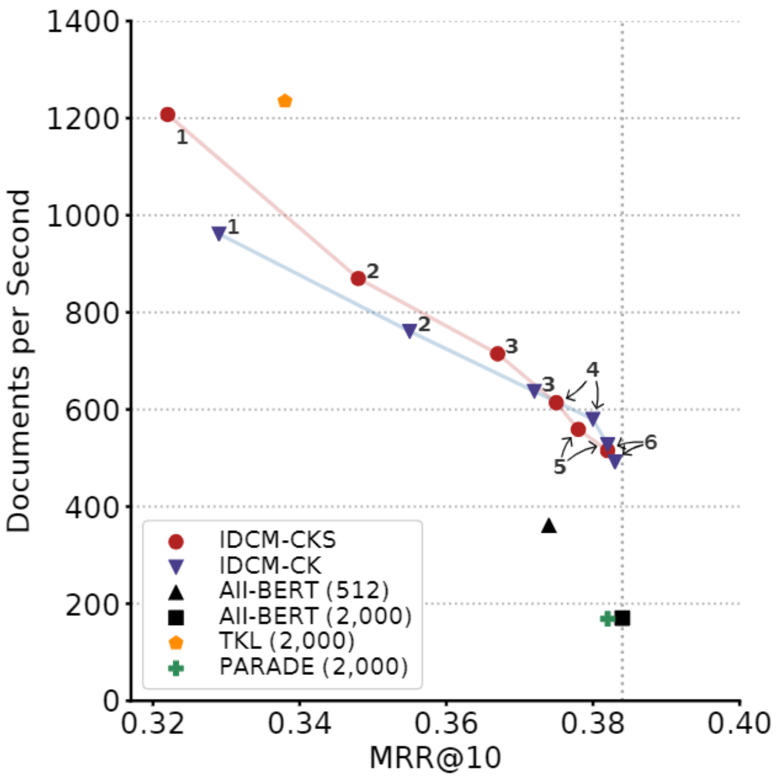
Sebastian Hofstätter, Hamed Zamani, Bhaskar Mitra, Nick Craswell and Allan Hanbury. 2021.
Local Self-Attention over Long Text for Efficient Document Retrieval. In SIGIR.

IDCM: Distilled Training



- IDCM is trained in 3 steps:
  - 1. & 2. Prepare slow, but effective BERT model
  - 3. Distill BERT passage-ranks into fast student (CK)
- We want a slow model, that gets the same passages in the top spot, for BERT to fine-grained score
  - Easier the more relevant a document is

IDCM: Improving Throughput



- We can choose different cascade settings:
  - How many passages to score with the costly model
  - The more we score, the better the results, but at the cost of efficiency
- IDCM offers you to choose exactly where you want to be on that curve
  - Ends with up to ~4x lower query latency, but same results

More Resources

- Pretrained Transformers for Text Ranking: BERT and Beyond
  Jimmy Lin et al https://arxiv.org/abs/2010.06467
  - Great Survey on different techniques using Transformers in Ranking
- ECIR 2021 Tutorial - IR From Bag-of-words to BERT and Beyond through Practical Experiments https://github.com/terrier-org/ecir2021tutorial
  - Hands on Tutorial with many Google Colab Notebooks
- Running list of new papers in the field: https://arxiv.org/list/cs.IR/recent
  - ~10 new papers a day; the fastest source to get info about pre-prints

Summary: Transformer Contextualized Re-Ranking

1. The concatenated BERT_{CAT} opened a new era of information retrieval
2. BERT provides enormous effectiveness jumps at the cost of speed
3. Combining Transformers and Kernels leads to a good compromise

1. The concatenated BERTCAT opened a new era of information retrieval
2. BERT provides enormous effectiveness jumps at the cost of speed
3. Combining Transformers and Kernels leads to a good compromise

Thank You