Improving Transformer-Kernel Ranking Model Using Conformer and Query Term Independence

Bhaskar Mitra[1], Sebastian Hofstätter[2], Hamed Zamani[3], Nick Craswell[1]
[1] Microsoft, [2] TU Wien, [3] University of Massachusetts Amherst
[1] {bmitra, nickcr}@microsoft.com, [2] s.hofstaetter@tuwien.ac.at, [3] zamani@cs.umass.edu

ABSTRACT

The Transformer-Kernel (TK) model has demonstrated strong reranking performance on the TREC Deep Learning benchmark—and can be considered to be an efficient (but slightly less effective) alternative to other Transformer-based architectures that employ (i) large-scale pretraining (high training cost), (ii) joint encoding of query and document (high inference cost), and (iii) larger number of Transformer layers (both high training and high inference costs). Since, a variant of the TK model—called TKL—has been developed that incorporates local self-attention to efficiently process longer input sequences in the context of document ranking. In this work, we propose a novel Conformer layer as an alternative approach to scale TK to longer input sequences. Furthermore, we incorporate query term independence and explicit term matching to extend the model to the full retrieval setting. We benchmark our models under the strictly blind evaluation setting of the TREC 2020 Deep Learning track and find that our proposed architecture changes lead to improved retrieval quality over TKL. Our best model also outperforms all non-neural runs ("trad") and two-thirds of the pretrained Transformer-based runs ("nnlm") on NDCG@10.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**; **Evaluation of retrieval results**; • **Computing methodologies → Neural networks**.

KEYWORDS

Deep learning; document ranking; efficiency

ACM Reference Format:
Bhaskar Mitra, Sebastian Hofstätter, Hamed Zamani, Nick Craswell. 2021. Improving Transformer-Kernel Ranking Model Using Conformer and Query Term Independence. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3404835.3463049

1 INTRODUCTION

In the inaugural year of the TREC Deep Learning track [13], ranking models using Transformers [59] demonstrated substantial improvements over traditional information retrieval (IR) methods [10]. Several of these approaches—*e.g.*, [64, 67]—employ BERT [19], with large-scale pretraining, as their core architecture. Diverging from this trend, Hofstätter et al. [23] propose the Transformer-Kernel (TK) model with few key distinctions: (i) TK uses a shallower model with only two Transformer layers, (ii) there are no computation-intensive pretraining, and (iii) TK independently encodes the query and document allowing for offline precomputations for faster response times. Consequently, TK achieves competitive performance at a fraction of the training and inference cost of its BERT-based peers.

Notwithstanding these efficiency gains, the TK model shares two critical drawbacks with other Transformer-based models. Firstly, the memory complexity of the self-attention layers is quadratic $O(n^2)$ with respect to the length n of the input sequence. This restricts the number of document terms we can inspect under fixed GPU memory budget. A trivial workaround involves inspecting only the first k terms of the document. This approach can negatively impact retrieval quality and has been shown to under-retrieve longer documents [22]. Secondly, in any real IR system, it is impractical to exhaustively evaluate every document in the collection for every query—and therefore these systems typically enforce some sparsity property to drastically narrow down the set of candidates. TK employs a nonlinear matching function over query-document pairs which makes it difficult to enforce such sparsity before model inference. This restricts TK's scope of application to late stage reranking of smaller candidate sets as identified by simpler retrieval models. So, in this work, we extend TK in the following ways:

(1) To scale to long text, we replace the Transformer layers with novel Conformer layers whose memory complexity is $O(n \times d_{\text{key}})$, instead of $O(n^2)$,
(2) To enable fast retrieval with TK, we incorporate query term independence (QTI) [44], and finally,
(3) we complement TK's latent matching with lexical term matching as suggested previously by Mitra et al. [42, 43], which is known to be effective for full retrieval [21, 30, 43, 62].

We study the impact of aforementioned changes under the strictly-blind evaluation setting of the TREC 2020 Deep Learning track.

2 RELATED WORK

Scaling self-attention to long text. The self-attention layer, as proposed by Vaswani et al. [59], can be described as follows:

$$\text{Self-Attention}(Q,K,V) = \Phi\left(\frac{QK^\intercal}{\sqrt{d_k}}\right) \cdot V \tag{1}$$

Where, $Q \in \mathbb{R}^{n \times d_{\text{key}}}$, $K \in \mathbb{R}^{n \times d_{\text{key}}}$, and $V \in \mathbb{R}^{n \times d_{\text{value}}}$ are the query, key, and value matrices—and d_{key} and d_{value} are the dimensions of the key and value embeddings, respectively. Here, n is the length of the input sequence and Φ denotes a softmax along the last tensor dimension. The quadratic $O(n^2)$ memory complexity of self-attention is a direct consequence of the component $QK^\intercal$ that produces a $n \times n$ matrix. Recently, several approaches have been proposed to mitigate

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3463049

this quadratic complexity that broadly fall under: (i) Restricting self-attention to smaller local windows over the input [17, 51, 56, 65], or (ii) operating under the assumption that the attention matrix is low rank r [29, 54, 57, 61] and hence finding alternatives to explicitly computing the $QK^\intercal$ matrix, or (iii) hybrid approaches [3, 7, 63]. In IR, recently Hofstätter et al. [22] extended TK to longer text using local self-attention. Other more general approaches to reducing the memory footprint, such as model parallelization [55] and gradient checkpointing [3] have also been explored.

Full retrieval with deep models. Efficient retrieval using deep models is an important challenge in IR [38, 39]. One approach involves the dual encoder architecture where the query and document are encoded independently, and efficient retrieval is achieved by approximate nearest-neighbour search [1, 5, 27, 28, 31] or by employing inverted-index over latent representations [68]. Precise matching of terms or concepts may be difficult using query-independent latent document representations [32], and therefore these models are often combined with explicit term matching [42, 45].

An alternative approach assumes QTI in the design of the neural ranking model [44]. In these models, the estimated relevance score $S_{q,d} = \sum_{t \in q} s_{t,d}$ is the sum of the document scores *w.r.t.* individual query terms. Readers should note that QTI is already baked into several classical IR models, like BM25 [52]. Relevance models with QTI can be used to offline precompute all term-document scores, and subsequently efficient search is performed using inverted-index. Several recent neural IR models [14–16, 34, 35, 44] that incorporate QTI have obtained promising results under the full retrieval setting. Document expansion based methods [48, 50], using large neural language models, can also be classified as part of this approach, assuming the subsequent retrieval step employs a traditional QTI model like BM25. In all these cases, the focus of the deep model is to estimate the relevance of the document *w.r.t.* individual terms in the vocabulary that can be precomputed during indexing. Another approach may involve neural query reformulation [33, 47, 58], although these methods typically underperform compared to the methods considered here.

3 CONFORMER-KERNEL WITH QTI

Conformer. The quadratic memory complexity of self-attention layers *w.r.t.* the input length is a direct result of explicitly computing the attention matrix $QK^\intercal \in \mathbb{R}^{n \times n}$. In this work, we propose a new separable self-attention layer that avoids instantiating the full term-term attention matrix.

$$\text{Separable-Self-Attention}(Q,K,V) = \Phi(Q) \cdot A \tag{2}$$

Where, $A = \Phi(K^\intercal) \cdot V$. As previously, Φ denotes softmax along the last dimension of the input tensor. Note that, however, in this separable self-attention mechanism, the softmax operation is employed twice: (i) $\Phi(Q)$ computes the softmax along the d_{key} dimension, and (ii) $\Phi(K^\intercal)$ computes the softmax along the n dimension. By computing $A \in \mathbb{R}^{d_{\text{key}} \times d_{\text{value}}}$ first, we avoid explicitly computing the full term-term attention matrix. The memory complexity of the separable self-attention layer is $O(n \times d_{\text{key}})$, which is a significant improvement when $d_{\text{key}} \ll n$. We modify the standard Transformer block as follows: (i) We replace the standard self-attention layer with the more memory efficient separable self-attention layer, and (ii) we apply grouped convolution before the separable self-attention layers to better capture the local context based on the window of neighbouring terms. We refer to this combination of grouped convolution and Transformer with separable self-attention as a Conformer. We incorporate Conformers into TK as a direct replacement for the Transformer layers and name the new architecture as a Conformer-Kernel (CK) model. In relation to handling long input sequences, we also replace the standard Kernel-Pooling with windowed Kernel-Pooling [22] in our proposed architecture.

Query term independence. To incorporate QTI into CK, we make two simple modifications. Firstly, we simplify the query encoder by getting rid of the Transformer layers and only considering the non-contextualized embeddings for the query terms. Secondly, instead of applying the aggregation function over the full interaction matrix, we apply it to each row individually, which corresponds to individual query terms. The scalar outputs from the aggregation function are linearly combined to produce the final query-document score. Fig 1b shows the proposed CK-QTI architecture.

Explicit term matching. We adopt the Duet [40–42, 46] framework wherein the term-document score is a linear combination of outputs from a latent and and an explicit matching models.

$$s_{t,d} = w_1 \cdot \text{BN}(s_{t,d}^{(\text{latent})}) + w_2 \cdot \text{BN}(s_{t,d}^{(\text{explicit})}) + b \tag{3}$$

Where, $\{w_1, w_2, b\}$ are learnable parameters and $\text{BN}(x) = (x - \mathbb{E}[x])/(\sqrt{\text{Var}[x]})$ denotes the BatchNorm operation [24]. We employ CK and define a new lexical matching function modeled on BM25 to compute $s_{t,d}^{(\text{latent})}$ and $s_{t,d}^{(\text{explicit})}$, respectively.

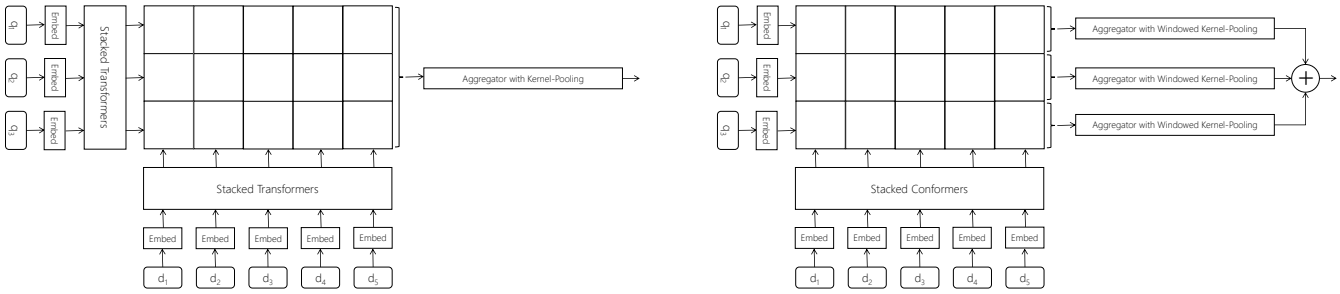
$$s_{t,d}^{(\text{explicit})} = \text{IDF}_t \cdot \frac{\text{BS}(\text{TF}_{t,d})}{\text{BS}(\text{TF}_{t,d}) + \text{ReLU}(w_{\text{dlen}} \cdot \text{BS}(|d|) + b_{\text{dlen}}) + \epsilon} \tag{4}$$

Where, IDF_t, $\text{TF}_{t,d}$, and $|d|$ denote the inverse-document frequency of the term t, the term-frequency of t in document d, and the length of the document, respectively. The w_{dlen} and b_{dlen} are the only two leanrable parameters of this submodel and ϵ is a small constant added to prevent a divide-by-zero error. The BatchScale (BS) operation is defined as $\text{BS}(x) = x/(\mathbb{E}[x] + \epsilon)$.

4 EXPERIMENT DESIGN

TREC 2020 Deep Learning Track. We evaluate CK under the strictly-blind TREC benchmarking setting by participating in the 2020 edition of the Deep Learning track [11], which: (a) provides stronger protection against overfitting that may result from the experimenter running multiple evaluations against the test set, and (b) is fairer to dramatically new approaches that may surface additional relevant documents not covered by pre-collected labels [66]. The 2020 track [11] uses the same training data as the previous year [10] originally derived from the MS MARCO dataset [2, 12]. However, the track provides a new blind test set for the second year. We only focus on the document ranking task and point the reader to [11] for further benchmarking details. We report NDCG@10 [25], NCG@100 [53], AP [69], and RR [8] against this blind set.

Model variants. We compare several variants of our model. The *NDRM1* variant incorporates Conformer layers and QTI into TK [23]. Figure 1 visualizes the NDRM1 architecture. The *NDRM2* model is a simple QTI-compliant explicit-term-matching model as described by Equation 4. A linear combination of NDRM1 and NDRM2 gives us the NDRM3 model. Because of the limit on the number of run submission to TREC, we only evaluate NDRM1 and NDRM3, although we confirm on the TREC 2019 test set that NDRM2 is competitive with a well-tuned BM25 baseline. The TREC 2020 Deep Learning track provided participants with a click log dataset called ORCAS [9]. We use clicked queries in the ORCAS data [9] as additional meta description



(a) Transformer-Kernel (TK) **(b) NDRM1 variant of Conformer-Kernel (CK) with QTI**

Figure 1: A comparison of the TK and the proposed CK-with-QTI architectures. In addition to replacing the Transformer layers with Conformers, the latter also simplifies the query encoding to non-contextualized term embedding lookup and incorporates a windowed Kernel-Pooling based aggregation that is employed independently per query term.

for corresponding documents to complement the intrinsic document content (URL, title, and body). Unlike previous work [68] on fielded document representations, we simply concatenate the different fields. We test each variant under both the rerank and the fullrank settings.

Model training. We consider the first 20 terms for every query and the first 4000 terms for every document. We pretrain the word embeddings using the word2vec [37] implementation in FastText [26]. We use a concatenation of the IN and OUT embeddings [43, 45] from word2vec to initialize the embedding layer parameters. The document encoder uses 2 Conformer layers and we set all hidden layer sizes to 256. We set the window size for the grouped convolution layers to 31 and the number of groups to 32. Correspondingly, we also set the number of attention heads to 32. We set the number of kernels k to 10. For windowed Kernel-Pooling, we set the window size to 300 and the stride to 100. Finally, we set the dropout rate to 0.2. For further details, please refer to the publicly released model implementation in PyTorch.[1] All models are trained on four Tesla P100 GPUs, with 16 GB memory each, using data parallelism.

We train the model using the RankNet objective [4]. For every positively labeled query-document pair in the training data, we randomly sample one negative document from the provided top 100 candidates corresponding to the query and two negative documents from the full collection. In addition to making pairs between the positively labeled document and the three negative documents, we also create pairs between the negative document sampled from the top 100 candidates and those sampled from the full collection, treating the former as more relevant. This can be interpreted as incorporating a form of weak supervision [18] as the candidates were previously generated using a traditional IR function.

5 RESULTS

RQ1. Does CK-QTI improve reranking quality over TKL? According to the taxonomy proposed by Craswell et al. [10], CK-QTI and TKL runs are the only "nn" runs—*i.e.*, neural models that do not use pretrained transformers—submitted to TREC 2020 Deep Learning track. TKL has previously been shown to outperform TK [22], and we confirmed with the submitting group that they considered these as well-tuned TKL runs. We also confirm that the related hyperparameters are comparable between the TKL runs and ours. Table 1 shows that in the same rerank setting, both NDRM1 and NDRM3 improve NDCG@10 over the best TKL run by 5.3%. The improvement from NDRM1 over TKL is statistically significant according to student's t-test ($p < 0.05$). However, similarly large improvement from NDRM3 over TKL is not stat. sig. likely due to small test set size. Even if we consider TK and CK to be comparable in results quality, the key motivation behind Conformers is their reduced GPU memory usage which we discuss next.

Table 1: Official TREC 2020 results. All metrics are computed at rank 100, except for NDCG which is computed at rank 10. Best and median runs are selected based on NDCG@10.

Run description	Subtask	NDCG	NCG	AP	RR
Other TREC runs for comparison					
Best "trad" run	fullrank	0.5629	0.6299	0.3829	0.9195
Best TKL run	rerank	0.5852	0.6283	0.3810	0.9296
Median "nnlm" run	fullrank	0.5907	0.6669	0.4259	0.8916
Best "nnlm" run	fullrank	0.6934	0.7718	0.5422	0.9476
Our models					
NDRM1	fullrank	0.5991	0.6280	0.3858	0.9333
NDRM1	rerank	0.6161	0.6283	0.4150	0.9333
NDRM3	rerank	0.6162	0.6283	0.4122	0.9333
NDRM3	fullrank	0.6162	0.6626	0.4069	0.9333
NDRM3 + ORCAS	rerank	0.6217	0.6283	0.4194	0.9241
NDRM3 + ORCAS	fullrank	0.6249	0.6764	0.4280	0.9444

RQ2. Does CK-QTI improve train-time GPU memory requirement over TKL? To demonstrate how the GPU memory consumption scales with respect to input sequence length, we plot the peak memory, across all four GPUs, for our proposed architecture using Transformer and Conformer layers, respectively, keeping all other hyperparameters and architecture choices fixed. Fig 2 shows the GPU memory requirement grows linearly with increasing sequence length for the Conformer, while quadratically when Transformers are employed. This is a significant improvement in GPU memory requirement over TK for longer text that could be further operationalized to improve training time convergence using larger batches or to incorporate longer input representations of documents.

RQ3. How does CK-QTI perform in the full retrieval setting? To enable retrieval from the full collection, we incorporate two

[1] https://github.com/bmitra-msft/TREC-Deep-Learning-Quick-Start

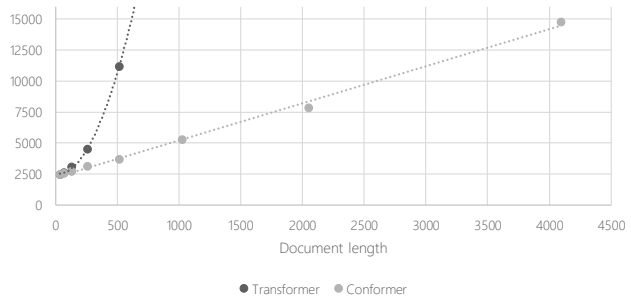
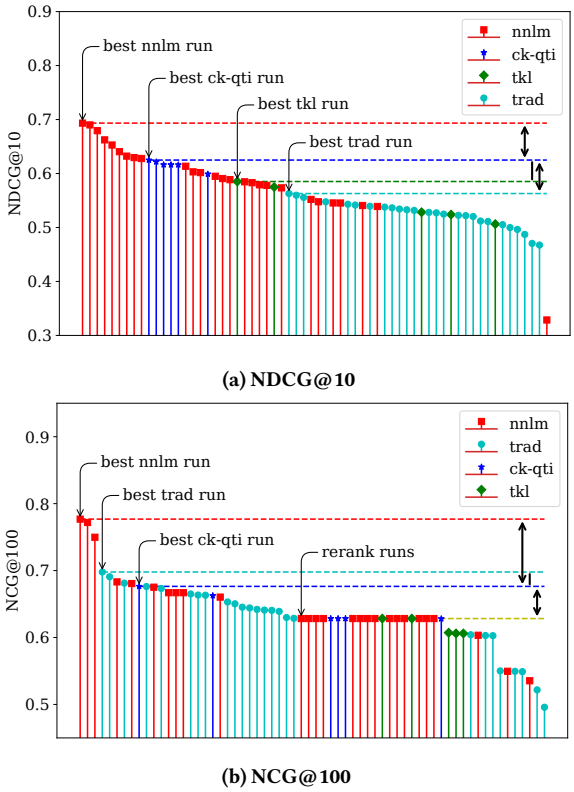


Figure 2: Comparison of peak GPU Memory Usage in MB, across all four GPUs, when employing Transformers *vs.* Conformers.

changes in TK: QTI and explicit term matching. QTI allows for precomputation of term-document scores and consequently fast retrieval using inverted-index data structures. The explicit term matching is expected to help with result quality under the full retrieval setting. In Table 1, we find that the NDRM3 variant—that incorporates explicit term matching—does indeed achieve 2.9% better NDCG@10 compared to the NDRM1 variant and 5.5% improvement in both AP and NCG@100. In contrast, both models achieve similar performance under the rerank setting. The candidate documents for reranking were generated by a first-stage BM25 ranker and hence explicit term matching signal is already part of this retrieval pipeline which may explain why we find no benefit from explicit term matching in reranking. These observations are supported by Kuzi et al. [30], who find that exact term matching are important for the fullrank setting. Also, NDRM1, in the absence of explicit term matching, achieves a lower NDCG@10 under the fullrank setting compared to the rerank setting. However, when explicit term matching is incorporated (*i.e.*, NDRM3), the metrics are comparable under both settings. Interestingly, when we include the ORCAS data in the document representation, we see improvements under the fullrank setting compared to reranking across all metrics: 2.2% for RR, 2.1% for AP, and 0.5% for NDCG@10. We confirm that the NDCG@10 improvement from fullrank over rerank setting under the NDRM3 + ORCAS configuration is stat. sig. based on a student's t-test ($p < 0.05$). Based on qualitative inspection of the queries, we find that exact term matching may be important for queries containing named entities—*e.g.*, "who is *aziz hashim*" and "why is *pete rose* banned from hall of fame"—where it is necessary to ensure that the retrieved documents are about the correct entity. Finally, with respect to the full retrieval setting, we note that NDRM3 with ORCAS improves NCG@100 by 7.7% over the provided candidates for the reranking setting, which puts it among the 10 top performing runs according to NCG@100 as seen in Fig 3.

RQ4. How does CK-QTI compare to "trad" and "nnlm" runs? In adhoc retrieval, a common strategy involves sequentially cascading multiple rank-and-prune stages [6, 20, 36, 49, 60] for better effectiveness-efficiency trade-offs. The multiple stages can improve result quality at additional computation costs. However, in our experiments under the full retrieval setting, we employ CK-QTI as a single stage retriever. Despite of this straightforward and efficient setup, we find that all three runs NDRM1, NDRM3, and NDRM3 + ORCAS achieve better NDCG@10 compared to the best non-neural (*i.e.*, "trad") run. The improvements from NDRM3, both with and without



(a) NDCG@10

(b) NCG@100

Figure 3: Comparing CK-QTI runs with runs submitted by other groups. The runs in each plot are sorted independently based on the corresponding metric.

the ORCAS-based document representation, is stat. sig. compared to the best "trad" run based on student's t-test ($p < 0.05$). Additionally, NDRM3, with and without ORCAS, outperforms two-thirds of the "nnlm" runs that employ costly pretraining of Transformers. The "nnlm" runs that outperform CK-QTI not only employ cascades of multiple rank-and-prune stages but sometimes multiple of those stages employ costly models like BERT. In contrast, CK-QTI retrieves from the full collection in one-shot and its performance can be likely improved by additional reranking stages.

6 CONCLUSION

We update TK by (i) replacing Transformers with Conformers, and incorporating (ii) QTI and (iii) explicit term matching. Conformers scale better to longer inputs and show both relevance and GPU memory improvements. Incorporating QTI and explicit term matching adapts the model to fullrank setting. In spite of being a single-stage retriever, CK-QTI outperforms all traditional methods and two-thirds of pretrained Transformer models. We believe that CK, like its predecessor TK, represents an alternative to BERT-based ranking models at lower training and run-time inference cost.

Acknowledgements. This work was supported in part by the Center for Intelligent Information Retrieval. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

REFERENCES

[1] Amin Ahmad, Noah Constant, Yinfei Yang, and Daniel Cer. 2019. ReQA: An evaluation for end-to-end answer retrieval models. *arXiv preprint arXiv:1907.04780* (2019).

[2] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv preprint arXiv:1611.09268* (2016).

[3] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *arXiv preprint arXiv:2004.05150* (2020).

[4] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proc. ICML*. ACM, 89–96.

[5] Wei-Cheng Chang, Felix X Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. *arXiv preprint arXiv:2002.03932* (2020).

[6] Ruey-Cheng Chen, Luke Gallagher, Roi Blanco, and J Shane Culpepper. 2017. Efficient Cost-Aware Cascade Ranking in Multi-Stage Retrieval. In *Proc. of SIGIR*.

[7] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating Long Sequences with Sparse Transformers. *CoRR* abs/1904.10509 (2019). arXiv:1904.10509 http://arxiv.org/abs/1904.10509

[8] Nick Craswell. 2009. Mean Reciprocal Rank. *Encyclopedia of database systems* 1703 (2009).

[9] Nick Craswell, Daniel Campos, Bhaskar Mitra, Emine Yilmaz, and Bodo Billerbeck. 2020. ORCAS: 18 Million Clicked Query-Document Pairs for Analyzing Search. *arXiv preprint arXiv:2006.05324* (2020).

[10] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2020. Overview of the TREC 2019 deep learning track. In *Proc. TREC*.

[11] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the TREC 2020 deep learning track. In *Proc. TREC*.

[12] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2021. MS MARCO: Benchmarking Ranking Models in the Large-Data Regime. In *Proc. SIGIR*. ACM (to appear).

[13] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Ellen Voorhees, and Ian Soboroff. 2021. TREC Deep Learning Track: Reusable Test Collections in the Large Data Regime. In *Proc. SIGIR*. ACM (to appear).

[14] Zhuyun Dai and Jamie Callan. [n.d.]. Context-Aware Passage Term Weighting For First Stage Retrieval. ([n. d.]).

[15] Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for IR with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 985–988.

[16] Zhuyun Dai and Jamie Callan. 2019. An Evaluation of Weakly-Supervised DeepCT in the TREC 2019 Deep Learning Track.. In *TREC*.

[17] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860* (2019).

[18] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W Bruce Croft. 2017. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 65–74.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[20] Luke Gallagher, Ruey-Cheng Chen, Roi Blanco, and J Shane Culpepper. 2019. Joint Optimization of Cascade Ranking Models. In *Proc. of WSDM*.

[21] Luyu Gao, Zhuyun Dai, Zhen Fan, and Jamie Callan. 2020. Complementing Lexical Retrieval with Semantic Residual Embedding. *arXiv preprint arXiv:2004.13969* (2020).

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

[23] Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2020. Interpretable & Time-Budget-Constrained Contextualization for Re-Ranking. In *Proc. of ECAI*.

[24] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[25] K. Järvelin and J. Kekäläinen. 2002. Cumulated Gain-Based Evaluation of IR Techniques. *ACM TOIS* 20, 4 (2002), 422–446.

[26] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* (2016).

[27] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. *arXiv preprint arXiv:2004.04906* (2020).

[28] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. *arXiv preprint arXiv:2004.12832* (2020).

[29] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2019. Reformer: The Efficient Transformer. In *International Conference on Learning Representations*.

[30] Saar Kuzi, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. 2020. Leveraging Semantic and Lexical Matching to Improve the Recall of Document Retrieval Systems: A Hybrid Approach. *arXiv preprint arXiv:2010.01195* (2020).

[31] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300* (2019).

[32] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2020. Sparse, Dense, and Attentional Representations for Text Retrieval. *arXiv preprint arXiv:2005.00181* (2020).

[33] Ji Ma, Ivan Korotkov, Yinfei Yang, Keith Hall, and Ryan McDonald. 2020. Zero-shot Neural Retrieval via Domain-targeted Synthetic Query Generation. *arXiv preprint arXiv:2004.14503* (2020).

[34] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Expansion via Prediction of Importance with Contextualization. *arXiv preprint arXiv:2004.14245* (2020).

[35] Joel Mackenzie, Zhuyun Dai, Luke Gallagher, and Jamie Callan. 2020. Efficiency Implications of Term Weighting for Passage Retrieval. In *Proceedings of the 43nd International ACM SIGIR Conference on Research & Development in Information Retrieval*.

[36] Irina Matveeva, Chris Burges, Timo Burkard, Andy Laucius, and Leon Wong. 2006. High Accuracy Retrieval with Multiple Nested Ranker. In *Proc. of SIGIR*.

[37] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. NIPS*. 3111–3119.

[38] Bhaskar Mitra. 2021. *Neural Methods for Effective, Efficient, and Exposure-Aware Information Retrieval*. Ph.D. Dissertation. University College London.

[39] Bhaskar Mitra and Nick Craswell. 2018. An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval* (2018).

[40] Bhaskar Mitra and Nick Craswell. 2019. Duet at TREC 2019 Deep Learning Track. In *Proc. TREC*.

[41] Bhaskar Mitra and Nick Craswell. 2019. An Updated Duet Model for Passage Re-ranking. *arXiv preprint arXiv:1903.07666* (2019).

[42] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match Using Local and Distributed Representations of Text for Web Search. In *Proc. WWW*. 1291–1299.

[43] Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. 2016. A Dual Embedding Space Model for Document Ranking. *arXiv preprint arXiv:1602.01137* (2016).

[44] Bhaskar Mitra, Corby Rosset, David Hawking, Nick Craswell, Fernando Diaz, and Emine Yilmaz. 2019. Incorporating Query Term Independence Assumption for Efficient Retrieval and Ranking using Deep Neural Networks (under review). In *Proc. ACL*.

[45] Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving Document Ranking with Dual Word Embeddings. In *Proc. WWW*.

[46] Federico Nanni, Bhaskar Mitra, Matt Magnusson, and Laura Dietz. 2017. Benchmark for complex answer retrieval. In *Proc. ICTIR*. ACM, 293–296.

[47] Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-Oriented Query Reformulation with Reinforcement Learning. 574–583.

[48] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to docTTTTTquery. *Online preprint* (2019).

[49] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-Stage Document Ranking with BERT. *arXiv preprint arXiv:1910.14424* (2019).

[50] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. *arXiv preprint arXiv:1904.08375* (2019).

[51] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. *arXiv preprint arXiv:1802.05751* (2018).

[52] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[53] Corby Rosset, Damien Jose, Gargi Ghosh, Bhaskar Mitra, and Saurabh Tiwary. 2018. Optimizing query evaluations using reinforcement learning for web search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1193–1196.

[54] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2020. Efficient Content-Based Sparse Attention with Routing Transformers. *arXiv preprint arXiv:2003.05997* (2020).

[55] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053* (2019).

[56] Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive attention span in transformers. *arXiv preprint arXiv:1905.07799* (2019).

[57] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. 2020. Sparse Sinkhorn Attention. *arXiv preprint arXiv:2002.11296* (2020).

[58] Christophe Van Gysel, Bhaskar Mitra, Matteo Venanzi, Roy Rosemarin, Grzegorz Kukla, Piotr Grudzien, and Nicola Cancedda. 2017. Reply With: Proactive Recommendation of Email Attachments. In *Proc. CIKM*.

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[60] Lidan Wang, Jimmy Lin, and Donald Metzler. 2011. A Cascade Ranking Model for Efficient Ranked Retrieval. In *Proc. of SIGIR*.

[61] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-Attention with Linear Complexity. *arXiv preprint arXiv:2006.04768* (2020).

[62] Marco Wrzalik and Dirk Krechel. 2020. CoRT: Complementary Rankings from Transformers. *arXiv preprint arXiv:2010.10252* (2020).

[63] Zhanghao Wu, Zhijian Liu, Ji Lin, Yujun Lin, and Song Han. 2020. Lite Transformer with Long-Short Range Attention. In *International Conference on Learning Representations (ICLR '20)*.
[64] Ming Yan, Chenliang Li, Chen Wu, Bin Bi, Wei Wang, Jiangnan Xia, and Luo Si. 2019. IDST at TREC 2019 Deep Learning Track: Deep Cascade Ranking with Generation-based Document Expansion and Pre-trained Language Modeling. In *TREC*.
[65] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. *arXiv preprint arXiv:1906.08237* (2019).
[66] Emine Yilmaz, Nick Craswell, Bhaskar Mitra, and Daniel Campos. 2020. On the Reliability of Test Collections for Evaluating Systems of Different Types. In *Proc. of SIGIR*.
[67] Zeynep Akkalyoncu Yilmaz, Shengjin Wang, and Jimmy Lin. 2019. H2oloo at TREC 2019: Combining Sentence and Document Evidence in the Deep Learning Track. In *TREC*.
[68] Hamed Zamani, Mostafa Dehghani, W Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 2018. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 497–506.
[69] Mu Zhu. 2004. Recall, Precision and Average Precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo* 2 (2004), 30.