Long Document Ranking with Query-Directed Sparse Transformer

Jyun-Yu Jiang[†], **Chenyan Xiong**[‡], **Chia-Jung Lee**[§] **and Wei Wang**[†]
[†]Department of Computer Science, University of California, Los Angeles, USA
[‡]Microsoft Research AI, Redmond, USA
[§]Amazon, Seattle, USA
{jyunyu,weiwang}@cs.ucla.edu, chenyan.xiong@microsoft.com, cjlee@amazon.com

Abstract

The computing cost of transformer self-attention often necessitates breaking long documents to fit in pretrained models in document ranking tasks. In this paper, we design Query-Directed Sparse attention that induces IR-axiomatic structures in transformer self-attention. Our model, QDS-Transformer, enforces the principle properties desired in ranking: local contextualization, hierarchical representation, and query-oriented proximity matching, while it also enjoys efficiency from sparsity. Experiments on one fully supervised and three few-shot TREC document ranking benchmarks demonstrate the consistent and robust advantage of QDS-Transformer over previous approaches, as they either retrofit long documents into BERT or use sparse attention without emphasizing IR principles. We further quantify the computing complexity and demonstrates that our sparse attention with TVM implementation is twice more efficient that the fully-connected self-attention. All source codes, trained model, and predictions of this work are available at https://github.com/hallogameboy/QDS-Transformer.

1 Introduction

Pre-trained Transformers such as BERT (Devlin et al., 2019) effectively transfer language understanding to better relevance estimation in many search ranking tasks (Nogueira and Cho, 2019; Nogueira et al., 2019; Yang et al., 2019). Nevertheless, the effectiveness comes at the quadratic cost $O(n^2)$ in computing complexity corresponds to the text length n, prohibiting its direct application to long documents. Prior work adopts quick workarounds such as document truncation or splitting-and-pooling to retrofit the document ranking task to pretrained transformers. Whilst there have been successes with careful architecture design, those bandit-solutions inevitably introduce information loss and create complicated system pipelines.

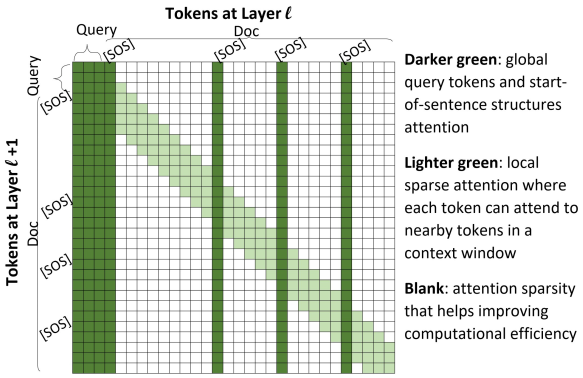
Tokens at Layer ℓ
Query
Doc
[SOS]
[SOS]
[SOS]
[SOS]
Query
[SOS]
Tokens at Layer ℓ +1
Doc
[SOS]
[SOS]
[SOS]
Darker green: global query tokens and start-of-sentence structures attention
Lighter green: local sparse attention where each token can attend to nearby tokens in a context window
Blank: attention sparsity that helps improving computational efficiency

Figure 1: An example illustration of the attention mechanism used in Query-Directed Sparse Transformer.

Intuitively, effective document ranking does not require fully connected self-attention between all query and document terms. The relevance matching between queries and documents often takes place at text segments as opposed to individual tokens (Callan, 1994; Jiang et al., 2019), suggesting that a document term may not need information thousands of words away (Metzler and Croft, 2005; Child et al., 2019), and that not all document terms are useful to calculate the relevance to the query (Xiong et al., 2017). The fully connected attention matrix includes many unlikely connections that create efficiency debt in computing, inference time, parameter size, and training convergence.

This paper presents Query-Directed Sparse Transformer (QDS-Transformer) for long document ranking. In contrast to retrofitted solutions, QDS-Transformer fundamentally considers the desirable properties for assessing relevance by focusing on attention paths that matter. Using sparse local attention (Child et al., 2019), our model removes unnecessary connections between distant

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 4594–4605
November 16 - 20, 2020. ©2020 Association for Computational Linguistics

document tokens. Using global attention upon sentence boundaries, our model further incorporates the hierarchical structures within documents. Last but not the least, we use global attention on all query terms that direct the focus to the relevance matches between query-document term pairs. These three attention patterns in our Query-Directed Sparse attention, as illustrated in Figure 1, permit global dissemination of IR-axiomatic information while keeping computation compact and essential.

In our experiments with TREC Deep Learning Track (Craswell et al., 2020) and three more few-shot document ranking benchmarks (Zhang et al., 2020), QDS-Transformer consistently improves the standard retrofitting BERT ranking baselines (e.g., max-pooling on paragraphs) by 5% NDCG. It also shows gains over more recent transformer architectures that induces various sparse structures, including Sparse Transformer, Longformer, and Transformer-XH, as they were not designed to incorporate the essential information required in document ranking. In the meantime, we also thoroughly quantify the efficiency improvement from our query-directed sparsity, showing that with TVM support (Chen et al., 2018), different sparse attention patterns lead to variant training and inference speed up, and in general QDS-Transformer enjoys 200%+ speed up compared to vanilla BERT on long documents.

Our visualization also shows interesting learned attention patterns in QDS-Transformer. Similar to the observation on BERT in NLP pipeline (Tenney et al., 2019), in lower QDS-Transformer levels, the attention focuses more on learning the local interactions and document hierarchies, while in higher layers the model focuses more on relevance matching with the query terms. We also show examples that QDS attention may center on the sole sentence that directly answers the query, or may span across several sentences that cover different aspects of the query, depending on the scope of the intent; this brings the advantage of better interpretability based on sparse attention.

2 Related Work

Neural models have demonstrated significant advances across various ranking tasks (Guo et al., 2019). Early approaches investigated diverse ways to capture relevance between queries and documents (Guo et al., 2016; Xiong et al., 2017; Dai et al., 2018; Hui et al., 2017). And recently the state-of-the-art in many text ranking tasks has been taken by BERT or other pretrained language models (Devlin et al., 2019; Nogueira et al., 2019; Nogueira and Cho, 2019; Dai and Callan, 2019; Yang et al., 2019; Craswell et al., 2020), when sufficient relevance labels are available for fine-tuning (e.g., on MS MARCO (Bajaj et al., 2016)).

The improved effectiveness comes with the cost of computing efficiency with deep pretrained transformers, especially on long documents. This stimulates studies investigating ways to retrofit long documents to BERT's maximum sequence length limits (512). A vanilla strategy is to truncate or split the documents: Dai and Callan (2019) applied BERT ranking on each passage segmented from the document independently and explored different ways to combine the passage ranking scores, using the score of the first passage (BERT-FirstP), the best passage (BERT-MaxP) (also studied in Yan et al. (2020)), or the sum of all passage scores (BERT-SumP).

More sophisticated approaches have also been developed to introduce structures to transformer attentions. Transformer-XL employs recurrence on a sequence of text pieces (Dai et al., 2019), Transformer-XH (Zhao et al., 2020) models a group of text sequences by linking them with eXtra Hop attention paths, and Transformer Kernel Long (TKL) (Hofstätter et al., 2020) uses a sliding window over the document terms and matches them with the query terms using matching kernels (Xiong et al., 2017).

On the efficiency front, Kitaev et al. (2020) proposed Reformer that employed locality-sensitive hashing and reversible residual layers to improve the efficiency of Transformers. Child et al. (2019) introduced sparse transformers to reduce the quadratic complexity to $O(L\sqrt{L})$ by applying sparse factorizations to the attention matrix, making the use of self-attention possible for extremely long sequences. Subsequent work (Sukhbaatar et al., 2019; Correia et al., 2019) leverage a similar idea in a more adaptive way. Combining local windowed attention with a task motivated global attention, Beltagy et al. (2020) presented Longformer with an attention mechanism that scales linearly with sequence length.

3 Preliminaries on Document Ranking

Given a query q and a set of candidate documents $D = \{d\}$, the document ranking task is to produce the ranking score $f(q, d)$ for each candidate

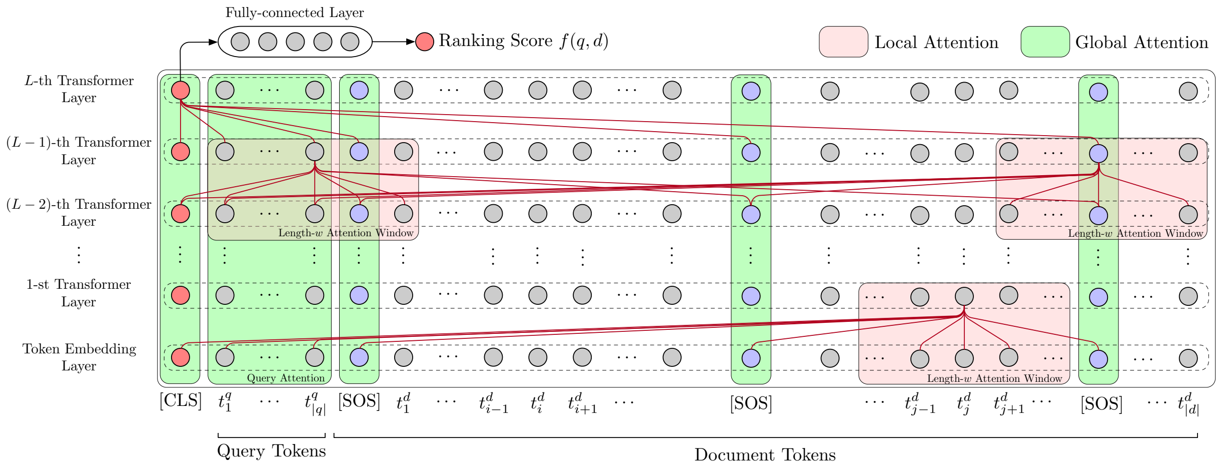
Fully-connected Layer
Ranking Score $f(q,d)$
Local Attention
Global Attention
L-th Transformer Layer
$(L-1)$-th Transformer Layer
$(L-2)$-th Transformer Layer
1-st Transformer Layer
Token Embedding Layer
Length-w Attention Window
Length-w Attention Window
Length-w Attention Window
Query Attention
[CLS] t_1^q $\cdots$ $t_{|q|}^q$ [SOS] t_1^d $\cdots$ t_{i-1}^d t_i^d t_{i+1}^d $\cdots$ [SOS] $\cdots$ t_{j-1}^d t_j^d t_{j+1}^d $\cdots$ [SOS] $\cdots$ $t_{|d|}^d$
Query Tokens
Document Tokens

Figure 2: The overall schema of our proposed QDS-Transformer.

document based on their relevance to the query.

BERT Ranker. The standard way to leverage pretrained BERT in document ranking is to concatenate the query and the document into one text sequence, feed it into BERT layers, and then use a linear layer on top of the last layer's [CLS] token (Nogueira and Cho, 2019):

$$f(q,d) = \text{Linear}(\text{BERT}([\text{CLS}] \circ q \circ [\text{SEP}] \circ d)).$$

This BERT ranker can be fine-tuned using relevance labels on (q,d) pairs, as simple as a classification task, and has achieved strong performances in various text ranking benchmarks (Bajaj et al., 2016; Craswell et al., 2020).

Transformer Layer. More specifically, let $\{t_0, t_1, ..., t_i, ..., t_n\}$ be the tokens in the concatenated q-d sequence, with query tokens $t_{1:|q|} \in q$ and document tokens $t_{|q|+1:n} \in |d|$, considering special tokens being part of q or d. The l-th transformer layer in BERT takes the hidden representations of previous layer (H^{l-1}), which is embedding for $l = 1$, and produces a new H^l as follows (Vaswani et al., 2017).

$$H^l = W^F(\hat{H}^l), \tag{1}$$

$$\hat{H}^l = A \cdot M \cdot V^T, \tag{2}$$

$$A = \mathbf{1}, \tag{3}$$

$$M = \text{softmax}(\frac{Q \cdot K^T}{\sqrt{d_k}}), \tag{4}$$

$$(Q^T; K^T; V^T) = (W^q; W^k; W^v) \cdot H^{l-1}. \tag{5}$$

It first passes the previous representations through the self-attention mechanism, using three projections (Eqn. 5), and then calculates the attention matrix between all token pairs using their query-key similarity (Eqn. 4, as in single-head formation). The attention matrix M then is used to fuse all other tokens' representation V, to obtain the updated representation for each position (Eqn. 2). In the end, another feed-foreword layer is used to obtain the final representation of this layer H^l (Eqn. 1).

The matrix A is the n^2 "adjencency" matrix in which each entry is one if there is an attention path between corresponding positions: $A_{ij} = 1$ means t_i queries the value of t_j using the key of t_j. In standard transformer and BERT, the attention paths are fully connected thus $A = \mathbf{1}$.

Computation Complexity. In each of the BERT layers, all the feed-forward operations (Eqn. 1 and 5) are applied to each individual token, leading to linear complexity w.r.t. text length n and the square of the hidden dimension size dim. The self-attention operation in Eqn. 2 and 4 calculates the attention strengths upon all token pairs, leading to squared complexity w.r.t text length but linear of the hidden dimension size.

The complexity of one transformer layer in BERT thus includes two components:

$$\underbrace{\mathcal{O}(\text{dim}^2 n)}_{\text{Feedforward}} + \underbrace{\mathcal{O}(n^2 \text{dim})}_{\text{Self-Attention}}. \tag{6}$$

The hidden dimension size (dim) is 768 in BERT Base and 1024 in BERT Large (Devlin et al., 2019). When the text sequence is longer than 1000 or 2000 tokens, which is often the case in document ranking (Craswell et al., 2020), the self-attention part becomes the main bottleneck in both computation and GPU memory. This leads to various retrofitted solutions that adapted the document ranking tasks to standard BERT which takes at most 512 tokens per sequence (Dai and Callan, 2019; Yang et al., 2019; Yan et al., 2020; Nogueira et al., 2019).

4 QDS-Transformer

Recent research has shown that with sufficient training and fully-connected self-attention, BERT learns attention patterns that capture meaningful structures in language (Clark et al., 2019) or for specific tasks (Zhao et al., 2020). However, this is not yet the case in long document ranking as computing becomes the bottleneck.

This section first presents how we overcome this bottleneck by injecting IR-specific inductive bias as sparse attention patterns. Then we discuss the efficient implementation of sparse attention.

4.1 Query-Directed Sparse Attention

Mathematically, inducing sparsity in self-attention is to modify the attention adjacency matrix A by only keeping connections that are meaningful for the task. For document retrieval, we include two groups of informative connections as sparse adjacency matrices: *local attention* and *query-directed global* attention.

4.1.1 Local Attention

Intuitively, it is unlikely that a token needs to see another token thousands of positions away to learn its contextual representation, especially in the lower transformer layers which are more about syntactic and less about long-range dependencies (Tenney et al., 2019). We follow this intuition used in the Sparse Transformer (Child et al., 2019) and define the following local attention paths:

$$A_{\text{local}}[i,j] = 1, \text{iff } |i-j| \leq w/2. \tag{7}$$

It only allows a token to see another token in each transformer layer if the two are $w/2$ position away, with w the window size. The local attention serves as the backbone for many sparse transformer variations as it provides the basic local contextual information (Correia et al., 2019; Sukhbaatar et al., 2019; Beltagy et al., 2020).

4.1.2 Query-Directed Global Attention

The local attention itself does not fully capture the relevance matches between the query and documents. We introduce several query-directed attention patterns to incorporate inductive biases widely used in document representation and ranking.

Hierarchical Document Structures. A common intuition in document representation is to leverage the hierarchical structures within documents, for example, words, sentences, paragraphs, and sections, and compose them into hierarchical attention networks (Yang et al., 2016). We use a two-level word-sentence-document hierarchy and inject this hierarchical structure by adding fully connected attention paths to all the sentences.

Specifically, we first prepend a special token [SOS] (start-of-sentence) to each sentence in the document, and form the following attention connections:

$$A_{\text{sent}}[i,j] = 1, \text{ iff } t_j = \text{[SOS]}. \tag{8}$$

Matching with the Query. For retrieval tasks, arguably the most important principle is to capture the semantic matching between queries and documents. Inducing this information is as simple as adding dedicated attention paths on query terms:

$$A_{\text{query}}[i,j] = 1, \text{ iff } t_i \in q. \tag{9}$$

It allows each token to see all query terms so as to learn query-dependent representations.

4.2 Summary

The three attention patterns together form the query-directed attention in QDS-Transformer:

$$A_{\text{QDS}} = A_{\text{local}} \cup A_{\text{sent}} \cup A_{\text{query}} \cup A_{\text{[CLS]}}. \tag{10}$$

We also add the global attention between all other tokens and [CLS]. Keeping everything else standard in BERT and using this query-directed sparse attention (A_{QDS}) in place of the fully-connected self-attention (A), we obtain our QDS-Transformer architecture as illustrated in Figure 2.

Interestingly, QDS-Transformer also resembles various effective IR-Axioms developed in past decades. For example, in QDS attention, a query term mainly focuses on the [SOS] token through A_{Sent}, while the [SOS] token recaps the proximity (Callan, 1994) matches locally around it through A_{Local}. The local attention in the query part also resembles the effective phrase matches (Metzler and Croft, 2005) as the query term representations are contextualized using other query terms through A_{Local}.

4.3 Efficient Sparsity Implementation

Our query-directed sparse attention reduces the self-attention complexity from $\mathcal{O}(n^2\text{dim})$ to $\mathcal{O}(n \cdot \text{dim} \cdot (w + |q| + |s|))$, where the local window size w and query length $|q|$ are constant to document length, and the number of sentences is orders of magnitude smaller.

However, to implement this sparsity efficiently on GPU is not that straightforward. Naively using

	Ad-hoc	Few-shot (avg. over 5 folds)		
	TREC19 DL	RB04	CW09-B	CW12-B13
Train queries	367,013	150	120	60
Train qrels	384,597	186,846	28,278	17,343
Dev queries	5,193	50	40	20
Dev qrels	519,300	62,282	9,426	5,781
Test queries	43	50	40	20
Test qrels	16,258	62,282	9,426	5,781

Table 1: The statistics of the experimental datasets.

for-loops or masking the adjacency matrix A may result in even worse efficiency than the full self-attention in common deep learning frameworks. An efficient implementation of sparse operations often requires customized CUDA kernels, which are inconvenient and require expertise in low-level GPU operations (Child et al., 2019). Inspired by Longformer (Beltagy et al., 2020), we address this issue by implementing QDS-Transformer with Tensor Virtual Machine (TVM) (Chen et al., 2018). Precisely, we implement custom CUDA kernels using TVM to dynamically compile our attention map A_{QDS} into efficiency-optimized CUDA codes.

5 Experimental Methodologies

This section discusses our experimental settings.

TREC 2019 Deep Learning Track Benchmark. We evaluate QDS-Transformer based on the document ranking task from this recent TREC benchmark (Craswell et al., 2020), specifically using the reranking subtask to rerank top-100 BM25 retrieved documents. The official evaluation metric is NDCG@10 on the test set. We also report MAP on test and MRR@10 on the development set.

Few-shot Document Ranking Benchmarks. We then evaluate the generalization ability of QDS-Transformer in the few-shot setting (Zhang et al., 2020) using TREC datasets Robust04 (RB04), ClueWeb09-B (CW09), and ClueWeb12-B13 (CW12), in which labels are much fewer than DL track. Our experimental settings are consistent with prior work (Zhang et al., 2020) in using the“MS MARCO Human Labels”. Specifically, neural rankers trained with MARCO labels are used as feature extractors to enrich TREC documents, which are then tested with five-fold cross-validation (Dai et al., 2018).

Table 1 summarizes the statistics of four datasets. We describe more details about datasets and experimental settings in Appendix A.1.

Baselines. Our baselines include multiple neural IR models and the best official TREC runs of single models. The main baselines cover:

- Relying on BERT models, RoBERTa (FirstP) only considers the first paragraph, while RoBERTa (MaxP) encodes short paragraphs with BERT and combines them with a max-pooling layer (Dai and Callan, 2019).
- Transformer-XH (Zhao et al., 2020) retrofits data pipelines to create independent sentences which are fed into BERT models, and aggregates them with an extra-hop attention layer.
- TK (Hofstätter et al., 2020) and TKL (Hofstätter et al., 2020) apply BERT-based kernels to estimate the relevance over document tokens with full attention.
- Sparse-Transformer (Child et al., 2019) applies length-w sparse local attention windows without considering query tokens.
- Longformer also uses sparse local attention and adds global attention by prepending one special token respectively to the query and document, same as in their (Beltagy et al., 2020) QA setup.

For ad-hoc retrieval, we also consider CO-PACRR (Hui et al., 2018) which employs CNNs without using pretrained NLM (non-PLM). Note that IDST (Yan et al., 2020) is not comparable because it exploits external generators for document expansion. For the few-shot learning task, we additionally compare with SDM, RankSVM, Coor-Ascent, and Conv-KNRM as reported in previous studies (Xiong et al., 2017; Dai et al., 2018). More details of the baselines can be found in Appendix B.

Implementation Details. We implement all methods with PyTorch (Paszke et al., 2019) and the Hugging Face transformer library (Wolf et al., 2019), excluding the baselines that have previously reported their scores. For sparse attention, we implement it using TVM with a custom CUDA kernel in PyTorch (Chen et al., 2018). Models are optimized by the Adam optimizer (Kingma and Ba, 2014) with a learning rate 10^{-5}, $(\beta_1, \beta_2) = (0.9, 0.999)$, and a dropout rate 0.1. The dev set is used for hyperparameter tuning to decide the best model, which is then applied to the test set. We set the maximum length of input sequences as 2,048. The dimension of the dense layer $\mathcal{F}_{\text{dense}}(\cdot)$ in relevance estimation is 768, while the local attention window size w is 128. All experiments are conducted on an Nvidia DGX-1 server with 512 GB memory and 8 Tesla V100 GPUs. Each method is limited to access only one GPU for fair comparisons.

TREC Deep Learning Track Document Ranking			
Method	Test Set		Dev Set
	NDCG@10	MAP	MRR@10
BM25	0.488	0.234	0.252
TREC Best Models			
BM25 (bm25tuned_prf)	0.528	0.386	0.318
Trad (srchvrs_run1)	0.561	0.349	0.306
Non-PLM (TUW19-d3-re)	0.644	0.271	0.401
BERT (bm25exp_marcomb)	0.646	0.424	0.352
Baseline Models			
CO-PACRR	0.550	0.231	0.284
TK	0.594	0.252	0.312
TKL	0.644	0.277	0.329
RoBERTa (FirstP)	0.588	0.233	0.278
RoBERTa (MaxP)	0.630	0.246	0.320
Sparse Attention based Models			
Sparse-Transformer	0.634	0.257	0.328
Longformer-QA	0.627	0.255	0.326
Transformer-XH	0.646	0.256	0.347
QDS-Transformer	**0.667**	**0.278**	**0.360**

Table 2: The ad-hoc retrieval performance of our approach and baseline methods on the TREC-19 DL track benchmark. Note that those baselines with higher MAP scores are all full retrieval and benefited from additional data engineering like query expansion.

6 Evaluation Results

This section evaluates QDS-Transformer in its effectiveness, attention patterns, and efficiency. We also analyze the learned query-directed attention weights and show case studies.

6.1 Retrieval Effectiveness

Table 2 summarizes the retrieval effectiveness on the TREC-19 DL benchmark. Table 3 shows the few-shot performance on the three TREC datasets.

QDS-Transformer consistently outperforms baseline methods on all datasets in both experimental settings. Note that the higher MAP scores from some methods in TREC-19 DL is because they have better first stage retrieval and are not using the same reranking setting. QDS-Transformer outperforms the best BERT-based TREC run by 3.25% in NDCG@10 and is more effective than the concurrent sliding window approach, TKL. Moreover, QDS-Transformer outperforms RoBERTa (MaxP), which is the standard retrofitted method for BERT, by 6% in NDGG@10 while also being a unified framework.

Compared with Sparse Transformers and Longformer-QA, QDS-Transformer provides more than 5% improvement in nearly all datasets. The best baseline is Transformer-XH, which creates structural sparsity by breaking a document into segments and introduces effective eXtra-hop attentions to jointly model the relevance of those segments. While these methods show competitive effectiveness especially with our TVM implementation, QDS-Transformer is consistently more accurate through the query-directed sparse attention patterns in all evaluation settings.

Method	**RB04**		**CW09**		**CW12**	
	NDCG	ERR	NDCG	ERR	NDCG	ERR
Classical IR; Cross Validated						
SDM	0.427	0.117	0.277	0.138	0.108	0.091
RankSVM	0.420	n.a.	0.289	n.a.	0.121	0.092
Coor-Ascent	0.427	n.a.	0.295	n.a.	0.121	0.095
Neural IR; Trained on MS MARCO and then Cross Validated.						
Conv-KNRM	0.427	0.117	0.287	0.160	0.112	0.092
RoBERTa (FirstP)	0.437	0.110	0.262	0.161	0.111	0.086
RoBERTa (MaxP)	0.439	0.114	0.264	0.162	0.092	0.074
Sparse-Transformer	0.449	0.119	0.274	0.173	0.119	0.094
Longformer-QA	0.448	0.113	0.276	0.179	0.111	0.085
Transformer-XH	0.450	0.123	0.283	0.179	0.107	0.080
QDS-Transformer	**0.457**	**0.126**	**0.308**	**0.191**	**0.131**	**0.112**

Table 3: The few-shot learning retrieval performance of different methods on three benchmark datasets. NDCG and ERR are at cut-off 20.

Method	Attention		TREC-19 DL Track	
	Q	Sent	NDCG@10	MAP
RoBERTa (MaxP)	✓	✗	0.630	0.246
Sparse Transformer	✗	✗	0.634	0.257
LongFormer-QA	✗	✗	0.627	0.255
Transformer-XH	✓	✓	0.646	0.256
QDS-Transformer (S)	✗	✓	0.633	0.244
QDS-Transformer (Q)	✓	✗	0.658	0.263
QDS-Transformer	✓	✓	0.667	0.278

Table 4: The retrieval performance of different models on the TREC-19 DL track benchmark dataset with different global attention patterns. Q and S indicate the usage of query and sentence global attention. Note that QDS-Transformer with no global attention is equivalent to Sparse-Transformer.

6.2 Effectiveness of Attention Patterns

This experiment studies the contribution of our query-directed sparse attention patterns to QDS-Transformer's effectiveness.

Table 4 shows the ablation results of the three attention patterns in TREC-19 DL benchmark: local attention only (A_{local}, Sparse Transformer), hierarchical attention on sentence only (A_{sent}, QDS-Transformer (S)), and query-oriented attention only (A_{query}, QDS-Transformer (Q)). All three sparse attention patterns contribute. As expected, query-oriented attention is most effective to capture the relevance match between query and documents. Note that the RoBERTa (MaxP) and Transformer-XH also attend to queries, but the attention is more localized as the document is broke into separated text pieces and the query is concatenated with each of them. In comparison, QDS-Transformer mimics the proximity matches and captures the global hier-

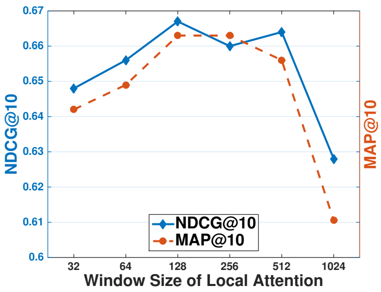

Figure 3: The performance of QDS-Transformer on TREC-19 DL track dataset with different local attention window sizes w.

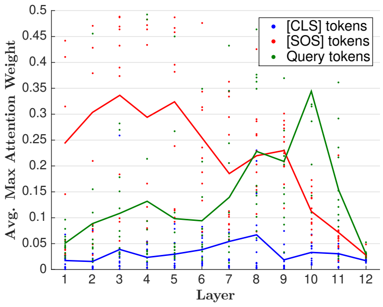

Figure 4: The average maximum attention scores to different types of tokens over Transformer layers.

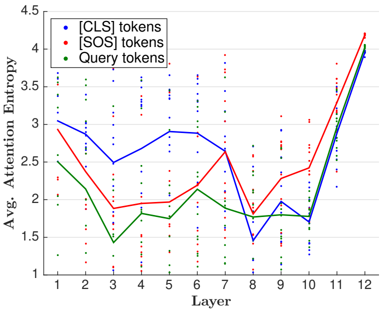

Figure 5: The average entropy scores of attention distributions for different token types over Transformer layers.

Method	Length	Sparsity	ms per q-d	
			Train	Infer
RoBERTa	1024	100%	391	100
RoBERTa	2048	100%	799	205
RoBERTa (FirstP)	512	100%	138	17
RoBERTa (MaxP)	4*512	25%	305	55
Transformer-XH	4*512	25%	309	54
QDS-Transformer (128)	512	30.84%	218	45
QDS-Transformer (128)	1024	18.72%	249	52
QDS-Transformer (128)	2048	8.97%	321	92
Longformer-QA (128)	2048	4.70%	166	45
Sparse-Transformer (128)	2048	4.56%	154	40
QDS-Transformer (32)	2048	6.70%	201	50
QDS-Transformer (64)	2048	8.97%	309	86
QDS-Transformer (128)	2048	13.53%	321	92
QDS-Transformer (256)	2048	22.64%	475	127
QDS-Transformer (512)	2048	40.88%	512	160
QDS-Transformer (1024)	2048	77.34%	629	195
QDS-Transformer (Q)	2048	5.10%	316	108
QDS-Transformer (S)	2048	8.57%	322	105
Without TVM Implementation				
Sparse-Transformer (128)	2048	4.56%	251	62
QDS-Transformer (128)	2048	13.53%	390	103

Table 5: Efficiency Quantification. The local attention window size is shown in parentheses. Q and S indicate the usage of only query and sentence attention. Sparsity is compared with fully attention at same text length.

archical structures in the document using dedicated attention from query terms to sentences.

Figure 3 depicts the change in retrieval effectiveness by varying the local attention window size. Both NDCG@10 and MAP@10 grow at a steady pace starting from a window size of 32 and peak at 128, but no additional gain is observed with bigger window sizes. The information from a term 512 tokens away does not provide many signals in relevance matching and is safely pruned in QDS-Transformer. Note that the dip at attention size 1024 is because our model is initialized from RoBERTa which is only pretrained on 512 tokens.

6.3 Model Efficiency

This experiment benchmarks the efficiency of different sparse attention patterns. Their training and inference time (ms per query-document pair, or MSpP) is shown in Table 5.

RoBERTa on 2048 tokens is prohibitive; we only measured its time with random parameters as we were not able to actually train it. Retrofitting was a natural choice to leverage pretrained models.

Sparsity helps. Sparse-Transformer (128) is much faster than MaxP. Interestingly, its attention matrix with only 4.56% non-zero entries leads to on par efficiency with retrofitted solutions and also only 5 times faster compared to full attention; this is due to the required cost involved in feed-forward. This effect is also reflected in the efficiency of QDS-Transformer with different local window sizes.

Different sparsity patterns dramatically influence the optimization of TVM. Intuitively, patterns with more regular shape would be easier to optimize than more customized connections in TVM. For example, the skipping patterns along sentence boundary in QDS-Transformer (S) seems more forgiving than the query-oriented attentions (Q). Comparing efficiency with and without our TVM implementation, the diagonal sparse shape in Sparse-Transformer is much better optimized.

How to better utilize the advantage from sparsity and structural inductive biases is perhaps a necessary future research direction in an era where models with fewer than one billion parameters are no longer considered large (Brown et al., 2020). Making progress in this direction may need more close collaborations between experts in application, modeling, and infrastructure.

Q1: 1037798 (who is robert gray) docid: D3533931	Q2: 1110199 (what is wifi vs bluetooth) docid: D1325409
Heads 1,2,4,6,9,10,11,12: Robert Gray (title)	Head 1: Bluetooth's low power consumption make it useful where power is limited.
	Head 2: Wi-Fi appliances are often plugged into wall outlets to operate.
Heads 3,5,7,8: Robert Gray, (born May 10, 1755, Tiverton, R.I. died summer 1806, at sea near eastern U.S. coast), captain of the first U.S. ship to circumnavigate the globe and explorer of the Columbia River.	Head 7: The extremely low power requirements of the latest Bluetooth 4.0 standard allows wireless connectivity to be added to devices powered only by watch batteries.
	Head 9: A Wi-Fi enabled network relies on a hub.
	Head 10: The advantages of using bluetooth from existing technology.
	Head 11: Wi-Fi is more suited to data-intensive activities such as streaming high-definition movies, while Bluetooth is better suited to tasks such as transferring keyboard strokes to a computer.
	Head 12: The greater power of Wi-Fi network also means it can move data more quickly than Bluetooth network.

Table 6: Case study of two queries on the sentences with the highest attention weights in the last transformer layer over different heads for the [CLS] token.

Q3: 1112341 (what is the daily life of thai people)	
Query Token	Sentence with the highest attention weight in the document D1641978
life	Children are expected to show great respect for their parents, and they maintain close ties, even well into adulthood .
thai	Culture of Thailand (title)

Table 7: Case study of the query 1112341 on the sentences in the document D1641978 with the highest attention weights among all heads from two query tokens. Note that we use attention weights in the third transformer layer.

6.4 Learned Attention Weights

This experiment analyzes the learned attention weights in QDS-Transformer, using the approach developed by Clark et al. (2019).

Figure 4 illustrates the average maximum attention weights of the three attention patterns used in our model. Interestingly, the model tends to implicitly conduct hierarchical attention learning (Yang et al., 2016), where lower layers focus on learning structures and pay more attention to [SOS] tokens, while higher layers emphasize the relevance by attending to queries more. Attention on both types of tokens is consistently stronger than on the [CLS] token. The model is capturing the inductive biases emphasized by our sparse attention structures.

Figure 5 shows the average entropy of the attention weight distribution. Intuitively, lower layer attention tends to have high entropy and thus a very broad view over many words, to create contextualized representations. The entropy of query and [SOS] are in general lower, as they focus on capturing information needs and document structures. The entropy of all three types of tokens rises again in the last layer, implying that they may try to aggregate representation for the whole input.

6.5 Case Study on Learned Attention Weights

Table 6 shows a case study of sentences with the highest attention weight from [CLS] in the last layer for two example queries. For factoid query Q1, all heads center on precise sentences that can directly answer the query. For Q2 that is on the exploratory side, different attention heads exhibit diverse patterns focusing on partial evidence that can provide a broader understanding collectively.

Table 7 depicts the other case study on learned attention weights of sentences from query tokens. We adopt the third transformer layer, where sentences obtain more attention as shown in Figure 4, to emphasize significant sentences for query tokens. The results show query-directed attention can capture sentences with different topics matched to individual query tokens, thereby comprehending sophisticated document structure.

These findings suggest that QDS-Transformer has an interesting potential to be applied to not only retrieval but also the question-answering task in NLP, providing a generic and effective framework, while also being interpretable with its the sparse structural attention connectivity. We further provide an additional case study in Appendix C.

7 Conclusions

QDS-Transformer improves the efficiency and effectiveness of pretrained transformers in long document ranking using sparse attention structures. The sparsity is designed to capture the principal properties (IR-Axioms) that are crucial for relevance modeling: local contextualization, document structures, and query-focused matching. In four TREC document ranking tasks with variant settings, QDS-Transformer consistently outperforms competitive baselines that retrofit to BERT or use sparse attention not designed for document ranking.

Our experiments demonstrate the promising future of joint optimization of structural domain knowledge and efficiency from sparsity, while its current form is somewhat at the infancy stage. Our analyses also indicate the potential of better interpretability from sparse structures and more unified models for IR and QA.

References

Zeynep Akkalyoncu Yilmaz, Shengjin Wang, and Jimmy Lin. 2019. H2oloo at trec 2019: Combining sentence and document evidence in the deep learning track. In *Proceedings of the Twenty-Eighth Text REtrieval Conference (TREC 2019)*.

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*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

James P Callan. 1994. Passage-level evidence in document retrieval. In *SIGIR'94*, pages 302–310. Springer.

Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. 2018. {TVM}: An automated end-to-end optimizing compiler for deep learning. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 578–594.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *URL https://openai.com/blog/sparse-transformers*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does bert look at? an analysis of bert's attention.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. In *EMNLP*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *arXiv preprint arXiv:2003.07820*.

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*, pages 985–988.

Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, page 126–134.

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. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM.

Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W. Bruce Croft, and Xueqi Cheng. 2019. A deep look into neural ranking models for information retrieval.

Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2019. Tu wien@ trec deep learning'19–simple contextualization for re-ranking. *arXiv preprint arXiv:1912.01385*.

Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2020. Interpretable & time-budget-constrained contextualization for re-ranking. *arXiv preprint arXiv:2002.01854*.

Sebastian Hofstätter, Hamed Zamani, Bhaskar Mitra, Nick Craswell, and Allan Hanbury. 2020. Local self-attention over long text for efficient document retrieval. In *SIGIR*.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard De Melo. 2018. Co-pacrr: A context-aware neural ir model for ad-hoc retrieval. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 279–287.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. Pacrr: A position-aware neural ir model for relevance matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058.

Jyun-Yu Jiang, Mingyang Zhang, Cheng Li, Michael Bendersky, Nadav Golbandi, and Marc Najork. 2019. Semantic text matching for long-form documents. In *The World Wide Web Conference*, pages 795–806.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451.*

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692.*

Donald Metzler and W Bruce Croft. 2005. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479.

Donald Metzler and W Bruce Croft. 2007. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085.*

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424.*

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive attention span in transformers. In *ACL*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950.*

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*, pages 5998–6008.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771.*

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval*, pages 55–64.

Ming Yan, Chenliang Li, Chen Wu, Bin Bi, Wei Wang, Jiangnan Xia, and Luo Si. 2020. Idst at trec 2019 deep learning track: Deep cascade ranking with generation-based document expansion and pre-trained language modeling.

Peilin Yang and Jimmy Lin. 2019. Reproducing and generalizing semantic term matching in axiomatic information retrieval. In *European Conference on Information Retrieval*, pages 369–381. Springer.

Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Simple applications of bert for ad hoc document retrieval. *arXiv preprint arXiv:1903.10972.*

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Kaitao Zhang, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. 2020. Selective weak supervision for neural information retrieval. In *Proceedings of The Web Conference 2020*, pages 474–485.

Chen Zhao, Chenyan Xiong, Corby Rosset, Xia Song, Paul Bennett, and Saurabh Tiwary. 2020. Transformer-xh: Multi-evidence reasoning with extra hop attention. In *International Conference on Learning Representations.*

Appendix

A Experimental Details

In this section, we clarify the details about experimental datasets and experimental settings.

A.1 Experimental Datasets

TREC-19 DL Track Dataset. For ad-hoc retrieval, we adopt the TREC-19 DL track benchmark as the experimental dataset with training, dev, and test sets. Training and dev sets consist of large-scale human relevance assessments derived from the MS MARCO collection (Bajaj et al., 2016) with no negative labels and sparse positive labels for each query while relevance judgments in the test sets are annotated by NIST judges.

Few-shot Document Ranking Benchmarks. For few-shot learning, three retrieval benchmark datasets are utilized in our experiments, including Robust04, ClueWeb09-B, and ClueWeb12-B13. Robust04 provides 249 queries from TREC Robust track 2014 with relevance labels. ClueWeb09-B includes of 200 queries with relevance labels from TREC Web Track 2009-2012. ClueWeb12-B13 consists of 100 queries from TREC Web Track 2013-2014 with relevance labels.

Note that Table 1 in the paper summarizes the statistics of four experimental datasets. Datasets of all benchmarks are publicly available. The TREC-19 DL track provides all dataset on its offical website[1]. The queries and relevance assessments of three few-shot document ranking datasets can be found at the TREC website[2] while document collocations are also publicly available on the corresponding sites[3][4][5].

A.2 Experimental Settings

Ad-hoc Retrieval. Experiments follow the protocol of the TREC-19 deep learning track. Each method is trained with the training set. The model parameters can be further fine-tuned with the dev set and the MRR@10 metric. The fine-tuned model is finally applied to the test set for evaluation. Following the official metrics, MRR@10 is used in dev set runs as labels are incomplete and shallow, while the test set is comprehensively evaluated using NDCG@10 and MAP@10.

Method	#Params	Method	#Params
RoBERTa (FirstP)	124M	RoBERTa (MaxP)	124M
Sparse-Transformer	149M	Longformer-QA	149M
Transformer-XH	128M	QDS-Transformer	149M

Table 8: Number of parameters for methods.

Few-shot Document Ranking. All experimental settings for few-shot learning are consistent with the"MS MARCO Human Labels" setting in previous studies (Zhang et al., 2020). Each method first trains a neural ranker on MARCO training labels, which are identical as in the TREC DL track. The latent representations of trained models are then considered as features for a Coor-Ascent ranker for low-label datasets using five-fold cross-validation (Dai and Callan, 2019; Dai et al., 2018) to rerank top-100 SDM retrieved results (Metzler and Croft, 2007). Standard metrics NDCG@20 and ERR@20 are used to compare the different approaches. The results are reported by taking the average of each test fold from the total 5 folds, wherein the rest 4 folds in each round are used as training and dev queries.

Hyperparameter Settings and Search. We adopt the pretrained model for sparse attention (Beltagy et al., 2020) and fix all of the hidden dimension numbers as 768 and the number of transformer layers as 12. BERT-based models use RoBERTa as pretrained models (Liu et al., 2019). To hyperparameter tuning, we search the local attention window size w in $\{32, 64, 128, 256, 512, 1024\}$ with the dev set and determine $w = 128$. Models are optimized by the Adam optimizer (Kingma and Ba, 2014) with a learning rate 10^{-5}, $(\beta_1, \beta_2) = (0.9, 0.999)$, and a dropout rate 0.1. Under the hyperparameter settings, the parameter numbers of our implemented methods are shown in Table 8 summarizing the sizes of parameters based on `model.parameters()` in PyTorch.

A.3 Evaluation Scripts

All evaluation measures are computed by the official scripts. For ad-hoc retreival, we use trec_eval[6] as the standard tool in the TREC community for evaluating ad-hoc retreival runs. This is also the official setting of the TREC-19 deep learning track. For few-shot document ranking, we use graded relevance assessment script (gdeval)[7] as the evaluation script measuring NDCG and ERR. Note that this setting is consistent with previous studies (Zhang et al., 2020; Dai and Callan, 2019).

[1] https://microsoft.github.io/TREC-2019-Deep-Learning/

[2] https://trec.nist.gov/

[3] RB04: https://trec.nist.gov/data/qa/T8_QAdata/disks4_5.html

[4] CW09: http://lemurproject.org/clueweb09/

[5] CW12: https://lemurproject.org/clueweb12/

[6] https://github.com/usnistgov/trec_eval

[7] https://trec.nist.gov/data/web/10/gdeval.pl

Sentence in the document D2944963 for Q4: 833860 (what is the most popular food in switzerland)	Top Query Token
Top 10 Swiss foods with recipes (title)	switzerland
You certainly won't go hungry in Switzerland.	food
You spear small cubes of bread onto long-stemmed forks and dip them into the hot cheese (taking care not to lose the bread in the fondue).	food
Jamie Oliver has this easy cheese fondue recipe, and this five-star recipe has good reviews.	popular

Table 9: Case study of the query 833860 with the query tokens with the highest attention weights in the 10-th transformer layer among all heads from the [SOS] tokens of sentences in the document D2944963.

B Baseline Methods

In this section, we introduce each baseline method.

TREC Best Runs.

- **bm25tuned_prf** (Yang and Lin, 2019) fine-tunes the BM25 parameters with pseudo relevance feedback as the best BM25 based method in official runs.
- **srchvrs_run1** is marked as the best traditional ranking method among official runs (Craswell et al., 2020).
- **TUW19-d3-re** (Hofstätter et al., 2019) as the best method without using non-pretrained language models (non-PLM) in official runs utilizes a transformer to encode both of the query and the document, thereby measuring interactions between terms and scoring the relevance.
- **bm25_expmarcomb** (Akkalyoncu Yilmaz et al., 2019) combines sentence-level and document-level relevance scores with a pre-trained BERT model.

Classical IR Methods.

- **SDM** (Metzler and Croft, 2005) as a sequential dependence model conducts ranking based on the theory of probabilistic graphical models. We obtain ranking results of SDM from previous studies (Dai and Callan, 2019). SDM is not only treated as a baseline method but also providing the candidate documents for reranking in the few-shot learning task.
- **Coor-Ascent** (Metzler and Croft, 2007) is a linear feature-based model for ranking. It is also considered as the trainer in few-shot learning with representations from methods.

Neural IR Methods.

- **CO-PACRR** (Hui et al., 2018) utilizes CNNs to model query-document similarity matrices and provide a score using a max-pooling layer.
- **Conv-KNRM** (Dai et al., 2018) applies CNNs to independently encode the query and the document. The encoded representations are then integrated by a cross-matching layer, thereby deriving relevance scores.

Transformer-based Methods.

- **TK** (Hofstätter et al., 2020) and **TKL** (Hofstätter et al., 2020) apply transformers to independently model the query and document, thereby measuring term interactions at the embedding level.
- **RoBERTa (FirstP)** and **RoBERTa (MaxP)** (Dai and Callan, 2019) adapt long-form documents by considering the first paragraph and combining RoBERTa outputs with max-pooling over paragraphs. Note that each paragraph is also attached with query tokens before being fed into the model.
- **Transformer-XH** (Zhao et al., 2020) encodes each sentence independently and considers their relations with an extra-hop attention layer. Note each sentence is also attached with query tokens as the model input.
- **Sparse-Transformer** (Child et al., 2019) simply uses sparse local attention to tackle the efficiency issue of transformers.
- **Longformer-QA** (Beltagy et al., 2020) extends Sparse-Transformer by attaching two global attention tokens to the query and the document as their settings for question answering. Note that their global attention would not consider document structural information.

C Additional Study on Attention Weights

In addition to attention from the classification token [CLS] and query tokens as shown in Section 6.5, here we analyze the attention from sentences. Table 9 shows the query tokens with the highest attention weights in the 10-th transformer layer among all head from the [SOS] tokens of sentences. Note that the 10-th transformer layer indicates higher importance of query tokens as shown in Figre 4. The results show that QDS-Transformer is capable of directing sentences to the tokens with matched topics, thereby understanding sophisticated document structure with different topics.