# Investigating the Effects of Sparse Attention on Cross-Encoders

Ferdinand Schlatt, Maik Fröbe, and Matthias Hagen

Friedrich-Schiller-Universität Jena

**Abstract** Cross-encoders are effective passage and document re-rankers but less efficient than other neural or classic retrieval models. A few previous studies have applied windowed self-attention to make cross-encoders more efficient. However, these studies did not investigate the potential and limits of different attention patterns or window sizes. We close this gap and systematically analyze how token interactions can be reduced without harming the re-ranking effectiveness. Experimenting with asymmetric attention and different window sizes, we find that the query tokens do not need to attend to the passage or document tokens for effective re-ranking and that very small window sizes suffice. In our experiments, even windows of 4 tokens still yield effectiveness on par with previous cross-encoders while reducing the memory requirements by at least 22% / 59% and being 1% / 43% faster at inference time for passages / documents. Our code is publicly available.[1]

**Keywords:** Cross-encoder · Re-ranking · Windowed attention · Cross-attention

## 1 Introduction

Pre-trained transformer-based language models (PLMs) are important components of modern retrieval and re-ranking pipelines as they help to mitigate the vocabulary mismatch problem of lexical systems [61, 84]. Especially cross-encoders are effective [52, 55, 57, 81] but less efficient than bi-encoders or other classic machine learning-based approaches with respect to inference run time, memory footprint, and energy consumption [68]. The run time issue is particularly problematic for practical applications as searchers often expect results after a few hundred milliseconds [2]. To increase the efficiency but maintain the effectiveness of cross-encoders, previous studies have, for instance, investigated reducing the number of token interactions by applying sparse attention patterns [44, 70].

Sparse attention PLMs restrict the attention of most tokens to local windows, thereby reducing token interactions and improving efficiency [72]. Which tokens have local attention is a task-specific decision. For instance, cross-encoders using the Longformer model [3] apply normal global attention to query tokens but local attention to document tokens. The underlying idea is that a document token does

[1]https://github.com/webis-de/ECIR-24

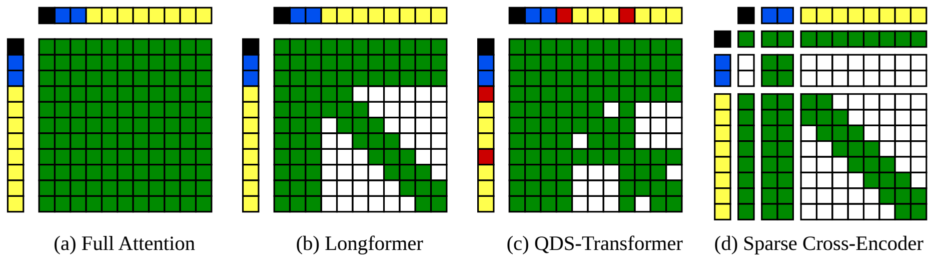


Figure 1: Previous cross-encoder attention patterns (a, b, and c) and our newly proposed sparse cross-encoder (d). The marginal boxes denote input tokens (black: [CLS], blue: query, yellow: passage / document, red: start of sentence). The inner green boxes indicate token attention. Our new pattern considers the sub-sequences separately (indicated by the added spacing) and is asymmetric.

not require the context of the entire document to determine whether it is relevant to a query—within a document, most token interactions are unnecessary, and a smaller context window suffices (cf. Figure 1 (b) for a visualization).

Previously, sparse attention has been applied to cross-encoders to be able to re-rank long documents without cropping or splitting them [44, 70]. However, the impact of sparsity on effectiveness has not been investigated in detail. To close this gap, we explore the limits of sparse cross-encoders and try to clarify which token interactions are (un)necessary. As mechanisms to reduce token interactions, we investigate local attention window sizes and disabling attention between sub-sequences (e.g., between the query and the passage or document). Our analyses are based on the following two assumptions.

(1) Cross-encoders create contextualized embeddings that encode query and passage or document semantics. We hypothesize that the contextualized embeddings of passage or document tokens do not actually need to encode fine-grained semantics. Rather, an overall "gist" in the form of small local context windows is sufficient to estimate relevance well.

(2) Cross-encoders allow queries and documents to exchange information via symmetric attention. We hypothesize that full symmetry is unnecessary as we view the query–document relevance relationship as asymmetric: for ranking, it should suffice to determine whether a result is relevant to a query, not vice versa. To further reduce token interactions, we propose a novel configurable asymmetric cross-attention pattern with varying amounts of interaction between [CLS], query, and passage or document tokens (cf. Figure 1 (d)).

In experiments on re-ranking tasks from the TREC Deep Learning tracks [26–29] and the TIREx benchmark [36], our new model's effectiveness is consistently on par with previous (sparse) cross-encoder models, even with a local window size of only four tokens and asymmetric attention. Further efficiency analyses for sequences with 174 / 4096 tokens show that our sparsification techniques reduce the memory requirements by at least 22% / 59% and yield inference times that are at least 1% / 43% faster.

## 2 Related Work

One strategy for using PLMs in ranking is to separate the encoding of queries and documents (or passages) [45, 46, 61, 67]. Such bi-encoder models are efficient retrievers as the document encodings can be indexed, so only the query needs to be encoded at retrieval time. However, the good efficiency of bi-encoders comes at reduced effectiveness compared to cross-encoders [40], especially in out-of-domain scenarios [66]. Consequently, bi-encoders are often used for first-stage retrieval, while the more effective but less efficient cross-encoders serve as second-stage re-rankers [52, 55, 57, 81]. We focus on improving the efficiency of cross-encoders while maintaining their effectiveness.

One strategy to make cross-encoders more efficient is to reduce the model size via knowledge distillation [41, 42, 50]. During knowledge distillation, a smaller and more efficient model aims to mimic a larger "teacher" model. The distilled models can often match the effectiveness of the teacher model at only a fraction of the computational costs [79], indicating that PLMs can be "overparameterized" for re-ranking [40]. We follow a similar idea and try to reduce token interactions in cross-encoders using sparse attention techniques to substantially lower computational costs at comparable effectiveness.

Previously, sparse attention PLMs aimed to increase the processable input length [72]. Instead of full attention across the entire input, a sparse PLM restricts attention to neighboring tokens. For instance, the Sparse Transformer [12] uses a block-sparse kernel, splitting the input into small blocks where tokens only attend to tokens in their block. Additional strided attention allows for attention between blocks. The Longformer [3], BigBird [83], and ETC [1] use a different approach. Tokens can attend to a fixed window of neighboring tokens, with additional global attention tokens that can attend to the entire sequence.

For efficient windowed self-attention, the Longformer, BigBird, and ETC use block-sparse matrices. However, block-sparse techniques often make concessions on the flexibility of window sizes or incur additional overhead that does not affect the resulting predictions. We compare several previously proposed windowed self-attention implementations and find them inefficient in terms of time or space compared to the reduction in operations. Therefore, we implement a custom CUDA kernel and compare it to other implementations (cf. Section 4.2).

Sparse attention PLMs have also been applied to document re-ranking. For example, the Longformer without global attention was used as a cross-encoder to re-rank long documents [70]. However, effectiveness was not convincing as later-appearing document tokens were unable to attend to query tokens. The QDS-Transformer [44] fixed this problem by correctly applying global attention to query tokens and achieved better retrieval effectiveness than previous cross-encoder strategies that split documents [30, 82]. While the QDS-Transformer was evaluated with different window sizes, the effectiveness results were inconclusive. A model fine-tuned on a window size of 64 tokens was tested with smaller (down to 16 tokens) and larger window sizes (up to 512)—always yielding worse effectiveness. We hypothesize that models specifically fine-tuned for smaller window sizes will be as effective as models fine-tuned for larger window sizes.

Besides analyzing models fine-tuned for different window sizes, we hypothesize that token interactions between some input sub-sequences are unnecessary. For example, bi-encoder models show that independent contextualization of query and document tokens can be effective [45, 46, 61, 67]. However, the symmetric attention mechanisms of previous sparse PLM architectures do not accommodate asymmetric attention patterns. We develop a new cross-encoder variant that combines windowed self-attention from sparse PLMs with asymmetric cross-attention. Cross-attention allows a sequence to attend to an arbitrary other sequence and is commonly used in transformer architectures for machine translation [73], computer vision [35, 59, 71], and in multi-modal settings [43].

## 3 Sparse Asymmetric Attention Using Cross-Encoders

We propose a novel sparse asymmetric attention pattern for re-ranking documents (and passages) with cross-encoders. Besides combining existing windowed self-attention and cross-attention ideas, our pattern also flexibly allows for asymmetric query–document interactions (e.g., allowing a document to attend to the query but not vice versa). To this end, we partition the input sequence into the [CLS] token, query tokens, and document tokens, with customizable attention between these groups and local attention windows around document tokens.

Figure 1 depicts our and previous cross-encoder attention patterns. In full attention, each token can attend to every other token. Instead, Longformer-based cross-encoders apply windowed self-attention to document tokens to which the QDS-Transformer adds global attention tokens per sentence. Our pattern is similar to the Longformer but deactivates attention from query tokens to [CLS] and document tokens. But, [CLS] and document tokens still have access to query tokens. Our hypothesis is that cross-encoders do not need symmetric query–document attention for re-ranking as a one-sided relationship suffices.

### 3.1 Preliminaries

A cross-encoder predicts a relevance score for a query–document pair $(q, d)$ as follows. Given $q$ and $d$ as token sequences $q_1 \dots q_m$ and $d_1 \dots d_n$ and adopting BERT-style encoding [33], a concatenation of a special classification token [CLS], then $q$, then a separator token [SEP], then $d$, and then another [SEP] token is passed through a transformer encoder. The output of the last transformer layer is an $s \times h$ matrix $O$, with $s = m + n + 3$ being the total sequence length and $h$ being the embedding dimensionality. Each column of $O$ is an embedding vector of a token from the input sequence. There is one such $O$-matrix per transformer layer, but only the [CLS] token embedding of the last transformer layer (first column of that layer's $O$-matrix) is used as input to a final linear transformation that then predicts the relevance score of $d$ for $q$.

The transformer encoder internally uses a dot-product attention mechanism [73]. For a single transformer layer, three separate linear transformations map the embedding matrix $O'$ of the previous layer to three vector-lookup matrices $Q$, $K$, and $V$. An $s \times s$ attention probability matrix that contains the probabilities of a token attending to another is obtained by softmaxing the $\sqrt{h}$-normalized product $QK^T$. The attention probabilities are then used as weights for the vector-lookup matrix $V$ to obtain the layer's output embedding matrix:

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

## 3.2 Windowed Self-Attention

Windowed self-attention was proposed for more efficient sparse PLM architectures [1, 3, 83]. The idea is that a token does not attend to the entire input sequence of length $s$, but only to a local window of $w$ tokens, with $w \ll s$ (e.g., $w = 4$ means that a token attends to $2 \cdot 4 + 1$ tokens: to the 4 tokens before itself, to itself, and to the 4 tokens after itself). For a window size $w$, windowed self-attention changes the dot-products of the transformer attention mechanism to windowed variants $\boxdot_w$ and $\odot_w$:

$$O = \text{Attention}_w(Q, K, V) = \text{softmax}\left(\frac{Q \boxdot_w K^T}{\sqrt{h}}\right) \odot_w V \,, \text{ with}$$

$$Q_{(s\times h)} \boxdot_w K^T_{(h\times s)} \rightarrow A_{(s\times 2w+1)}, \qquad \text{where } a_{i,j} = \sum_{l=1}^{h} q_{i,l} \cdot k_{l,i+j-w-1} \,,$$

$$P_{(s\times 2w+1)} \odot_w V_{(s\times h)} \rightarrow O_{(s\times h)}, \qquad \text{where } o_{i,l} = \sum_{j=1}^{2w+1} p_{i,j} \cdot v_{i+j-w-1,l} \,.$$

Thus, $\boxdot_w$ outputs a band matrix subset of the standard matrix–matrix multiplication, stored in a space-efficient form (non-band entries omitted), and $\odot_w$ multiplies a space-efficiently stored band matrix and a standard matrix. To ensure correctness, we zero-pad windows exceeding the sequence bounds: when either $i + j - w \leq 0$ or $i + j - w > s$, we set $k_{l,i+j-w-1} = v_{i+j-w-1,l} = 0$.

For a visual impression of windowed attention, consider the lower right document-to-document attention matrix in Figure 1 (d). Only the diagonal band is computed: in Figure 1 (d) for $w = 1$.

Compared to full self-attention, in theory, windowed self-attention reduces the space complexity from $\mathcal{O}(s^2)$ to $\mathcal{O}(s \cdot (2w + 1))$ and the (naïve) computational complexity from $\mathcal{O}(s^2 \cdot h)$ to $\mathcal{O}(s \cdot (2w + 1) \cdot h)$. However, fully achieving these improvements is difficult in practice. Previous windowed self-attention implementations avoided writing hardware-specific kernels and made concessions regarding flexibility, time efficiency, or space efficiency [3, 83]. Therefore, we implement our own CUDA kernel for windowed self-attention; Section 4.2 compares our implementation's efficiency to previous implementations.

## 3.3 Cross-Attention

Cross-attention is a type of attention where a token sequence does not attend to itself, as in self-attention, but to a different sequence. We use cross-attention to configure attention between different token types. Instead of representing a cross-encoder's input as a single sequence, we split it into three disjoint subsequences: the [CLS] token, the query tokens, and the document tokens (Figure 1 (d) visually represents this for our proposed pattern by splitting the marginal vectors; the [SEP] tokens are part of "their" respective subsequence). Each subsequence can then have its own individual attention function $\text{Attention}(Q, K, V)$.

We split the vector-lookup matrices column-wise into [CLS], query, and document token-specific submatrices $Q_c$, $Q_q$, $Q_d$, etc. These matrices are pre-computed and shared between the different attention functions for efficiency. Restricting attention between subsequences then means to call the attention function for a subsequence's $Q$-matrix and the respective $K$- and $V$-matrices of the attended-to subsequences. For example, to let a document attend to itself and the query, the function call is $\text{Attention}(Q_d, [K_q, K_d], [V_q, V_d])$, where $[\cdot,\cdot]$ denotes matrix concatenation by columns (i.e., $[M, M']$ yields a matrix whose "left" columns come from $M$ and the "right" columns from $M'$).

## 3.4 Locally Windowed Cross-Attention

The above-described cross-attention mechanism using concatenation is not directly applicable in our case, as we want to apply windowed self-attention to document tokens and asymmetric attention to query tokens. Instead of concatenating the matrices $K$ and $V$, our mechanism uses tuples $\mathcal{K}$ and $\mathcal{V}$ of matrices and a tuple $\mathcal{W}$ of window sizes to assign different attention window sizes $w \in \mathcal{W}$ to each attended-to subsequence. As a result, we can combine windowed self-attention with asymmetric attention based on token types. Formally, given $j$-tuples $\mathcal{K}$ and $\mathcal{V}$ of matrices $K_i$ and $V_i$ and a $j$-tuple $\mathcal{W}$ of window sizes $w_i$, our generalized windowed cross-attention mechanism works as follows:

$$\text{Attention}_{\mathcal{W}}(Q, \mathcal{K}, \mathcal{V}) \;=\; \sum_{i=1}^{j} P_i \odot_{w_i} V_i\,, \text{ where}$$

$$[P_1, \ldots, P_j] = \text{softmax}\left(\frac{[A_1, \ldots, A_j]}{\sqrt{h}}\right) \quad \text{and} \quad A_i = Q \boxdot_{w_i} K_i\,.$$

Our proposed attention pattern (visualization in Figure 1 (d)) is then formally defined as follows. The [CLS] token has full attention over all subsequences (Equation 1; for notation convenience, we use $w = \infty$ to refer to full self-attention), the query tokens can only attend to query tokens (Equation 2), and the document tokens can attend to all subsequences but use windowed self-attention on their own subsequence (Equation 3):

$$O_c = \text{Attention}_{(\infty,\infty,\infty)}(Q_c, (K_c, K_q, K_d), (V_c, V_q, V_d)), \tag{1}$$

$$O_q = \text{Attention}_{(\infty)}(Q_q, (K_q), (V_q)), \tag{2}$$

$$O_d = \text{Attention}_{(\infty,\infty,w)}(Q_d, (K_c, K_q, K_d), (V_c, V_q, V_d)). \tag{3}$$

### 3.5 Experimental Setup

We fine-tune various models using the Longformer and our proposed attention pattern with window sizes $w \in \{\infty, 64, 16, 4, 1, 0\}$ ($\infty$: full self-attention). We start from an already fine-tuned and distilled cross-encoder model[2] which also serves as our baseline [62]. We additionally fine-tune a QDS-Transformer model with its default $w = 64$ window for comparison. All models are fine-tuned for 100,000 steps with 1,000 linear warm-up steps and a batch size of 32 (16 document pairs) with margin MSE loss using MS MARCO-based knowledge distillation triples [40]. For documents, we extend the models fine-tuned on passages using positional interpolation [10] and further fine-tune them on document pairs from MS MARCO Document [54] for 20,000 steps using RankNet loss [8]. Negative documents are sampled from the top 200 documents retrieved by BM25 [65]. We use a learning rate of $7 \cdot 10^{-6}$, an AdamW optimizer [51], and a weight decay of 0.01. We truncate passages and documents to a maximum sequence length of 512 and 4096 tokens, respectively. All models were implemented in PyTorch [58] and PyTorchLightning [34] and fine-tuned on a single NVIDIA A100 40GB GPU.

We evaluate the models on the TREC 2019–2022 Deep Learning (DL) passage and document retrieval tasks [26–29] and the TIREx benchmark [36]. For each TREC DL task, we re-rank the top 100 passages / documents retrieved by BM25 using `pyserini` [49]. For TIREx, we use the official first-stage retrieval files retrieved by BM25 and ChatNoir [4, 60] and also re-rank the top 100 documents. We measure nDCG@10 and access all corpora and tasks via `ir_datasets` [53], using the default text field for passages and documents.

To evaluate time and space efficiency, we generate random data with a query length of 10 tokens and passage / document lengths from 54 to 4086 tokens. For the QDS-Transformer, we set global sentence attention at every 30th token, corresponding to the average sentence length in MS MARCO documents. We use the largest possible batch size per model, but up to a maximum of 100.

## 4 Empirical Evaluation

We compare our sparse cross-encoder's re-ranking effectiveness and efficiency to full attention and previous sparse cross-encoder implementations. We also examine the impact of different small window sizes and of our attention deactivation pattern—analyses that provide further insights into how cross-encoders work.

### 4.1 Effectiveness Results

*In-domain Effectiveness* Table 1 reports the nDCG@10 of various models with different attention patterns and window sizes on the TREC Deep Learning passage and document re-ranking tasks. We group Full Attention and Longformer models into the same category because they have the same pattern but different window sizes in our framework. We fine-tune separate models for passage and

[2]https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2

Table 1: Re-ranking effectiveness as nDCG@10 on TREC Deep Learning [26–29]. The highest score per task is given in bold. Scores obtained using a MaxP strategy are grayed out. † denotes significant equivalence within ±0.02 (paired TOST [69], $p < 0.003$), compared to Full Attention $w = \infty$ for passage tasks and Longformer $w = 64$ for document tasks.

| **Task** | | **Full Att. / Longformer** | | | | | | **Sparse Cross-Encoder** | | | | | | **QDS** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $w =$ | $\infty$ | 64 | 16 | 4 | 1 | 0 | $\infty$ | 64 | 16 | 4 | 1 | 0 | 64 |
| Passage | 2019 | .724 | .719† | .725† | .719 | .714 | .694 | .722 | .717 | .724 | **.728** | .715 | .696 | .720† |
| Passage | 2020 | .674 | .681† | .680 | **.684** | .676 | .632 | .666 | .672 | .661 | .665 | .649 | .605 | .682 |
| Passage | 2021 | **.656** | .653 | .650 | .645 | .629 | .602 | **.656** | .650 | .639 | .647 | .625 | .593 | **.656**† |
| Passage | 2022 | **.496** | .494† | .487 | .486 | .481 | .441 | .490 | .492† | .479 | .484 | .471 | .427 | .495† |
| | Avg. | .619 | .619† | .616† | .615† | .607 | .572 | .615† | .615† | .607 | .612† | .596 | .560 | **.620**† |
| Document | 2019 | .658 | .683 | .678 | .667 | .689 | .663 | .638 | .672 | .685 | .669 | .692 | .646 | **.697** |
| Document | 2020 | .622 | .640 | .639 | **.661** | .655 | .644 | .636 | .638 | .650 | .642 | .657 | .638 | .639 |
| Document | 2021 | .678 | .671 | .681 | **.683** | **.683** | .629 | .677 | .681 | .681 | .670 | .679 | .644 | .676 |
| Document | 2022 | .424 | .425 | .431 | .425 | .409 | .389 | .421 | **.446** | .443 | .417 | .424 | .405 | .428 |
| | Avg. | .575 | .582 | .586† | .587 | .584† | .556 | .573 | .590 | **.594** | .577 | .589 | .561 | .587† |

document re-ranking (cf. Section 3.5) except models with $w = \infty$. Their lack of efficiency prevents training on long sequences, and we only fine-tune them on passages but include their MaxP scores [30] for documents (in gray).

Since we hypothesize that sparse attention will not substantially affect the re-ranking effectiveness, we test for significant equivalence instead of differences. Therefore, we cannot use the typical t-test, but instead use a paired TOST procedure (two one-sided t-tests [69]; $p < 0.003$, multiple test correction [47]) to determine if the difference between two models is within ±0.02. We deem ±0.02 a reasonable threshold for equivalence since it is approximately the difference between the top two models in the different TREC Deep Learning tasks.

We consider two different reference models for the passage and document re-ranking tasks. The Full Attention cross-encoder has complete information access in the passage re-ranking setting and serves as the reference model for the passage tasks. Since the models without windowed attention ($w = \infty$) only process a limited number of tokens in the document re-ranking setting, we use the standard Longformer ($w = 64$) as the reference model for document tasks.

We first examine the effectiveness of the QDS-Transformer. In contrast to the original work [44], it does not improve re-ranking effectiveness despite having more token interactions. The reference models are statistically equivalent to the QDS-Transformer within ±0.02 for both passage and document re-ranking.

Next, we examine the effect of independent query contextualization on effectiveness. We compare the reference models for the passage and document tasks with our sparse cross-encoder with window size $w = \infty$ and $w = 64$, respectively. This comparison is a type of ablation test, as the two models being compared have identical configurations, except that our sparse cross-encoders independently contextualize the query. Our model is statistically equivalent within $\pm 0.02$ to the reference model on average across all passage tasks. The same does not hold for the document task, but our sparse cross-encoder is slightly more effective, achieving an 0.008 higher nDCG@10. We conclude that independent query contextualization only marginally affects re-ranking effectiveness.

Finally, we examine the effect of decreasing window size on effectiveness. For the Longformer, the window sizes 64, 16, and 4 are all significantly equivalent within $\pm 0.02$ on average across passage tasks. Even reducing the window size to just a single token, meaning a passage token can only attend to its immediate left and right neighboring tokens, reduces effectiveness by only 0.012 but is no longer statistically equivalent. The results are similar for the document tasks, with the window sizes 16 and 1 being statistically equivalent to the reference model. Furthermore, deactivating attention for passage or document tokens to its own subsequence ($w = 0$) does not yield statistically equivalent effectiveness but only drops effectiveness by 0.047 and 0.026 nDCG@10, respectively.

The effect of decreasing window sizes is similar for our sparse cross-encoder model. Window sizes 64 and 4 are statistically equivalent for passage tasks. Window sizes 16 and 1 are not statistically equivalent but only drop effectiveness by 0.012 and 0.023 nDCG@10, respectively. On the document tasks, our sparse cross-encoder models with smaller window sizes are never statistically equivalent within $\pm 0.02$ compared to the reference Longformer with window size 64. However, window sizes 64, 16, and 1 slightly improve over the reference model, and window size 4 is slightly less effective.

In summary, both independent query contextualization and windowed self-attention do not substantially affect re-ranking effectiveness, confirming our initial assumptions. That is, symmetric modeling of the query–passage relationship is unnecessary, and very small window sizes suffice to determine a passage's or document's relevance to the query. Interestingly, we find a window size of 0 to still feature competitive effectiveness. In this case, a document (or passage) token cannot attend to other tokens from its sub-sequence, making it similar to a lexical or bag-of-words model. We leave a more in-depth investigation into the implications of these results for future work.

*Out-of-domain Effectiveness* Table 2 reports nDCG@10 on all out-of-domain tasks from the TIREx [36] benchmark of our sparse cross-encoder compared to two other cross-encoders of various sizes: monoT5 [56] and monoBERT [55]. Our model uses a window size of 4 tokens, and all models use a maximum sequence length of 512 token except for our sparse cross-encoder trained on documents, which has access to a maximum of 4096 tokens. Overall, the out-of-domain re-ranking effectiveness of all cross-encoders is lower than in-domain. Only the monoT5 large and 3b variants and our sparse cross-encoder trained on passages can improve the ranking of the first-stage retrieval on average across all corpora.

Our model trained solely on passages (512-token sequence length) features competitive effectiveness despite having substantially fewer parameters. On average over all corpora, it slightly outperforms both the base and large monoBERT

Table 2: Re-ranking effectiveness as nDCG@10 on TIREx [36]. The average document length per corpus and first-stage (FS) effectiveness are listed for context. We report micro-averaged scores across all queries from a corpus and macro-average these in the "Average" row. The highest score per corpus is given in bold. Our sparse cross-encoder models use a window size of 4.

| **Corpus** | **Doc. Len.** | **FS** | **monoT5** | | | **monoBERT** | | **Sparse CE** | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Base | Large | 3b | Base | Large | 512 | 4096 |
| Antique [37] | 49.9 | .510 | .505 | .527 | .537 | .507 | .484 | **.540** | .174 |
| Args.me [5, 6] | 435.5 | **.405** | .305 | .338 | .392 | .314 | .371 | .313 | .180 |
| CW09 [14–17] | 1132.6 | .178 | .186 | .182 | .201 | .192 | .134 | .198 | **.212** |
| CW12 [5, 6, 21, 22] | 5641.7 | **.364** | .260 | .266 | .279 | .263 | .251 | .312 | .338 |
| CORD-19 [78] | 3647.7 | .586 | .688 | .636 | .603 | **.690** | .625 | .673 | .642 |
| Cranfield [19, 20] | 234.8 | .008 | .006 | .007 | .007 | .006 | .006 | **.009** | .003 |
| Disks4+5 [74–77] | 749.3 | .429 | .516 | .548 | **.555** | .514 | .494 | .487 | .293 |
| GOV [23–25] | 2700.5 | .266 | .320 | .327 | **.351** | .318 | .292 | .316 | .292 |
| GOV2 [9, 13, 18] | 2410.3 | .467 | .486 | .513 | **.514** | .489 | .474 | .503 | .460 |
| MED. [38, 39, 63, 64] | 309.1 | **.366** | .264 | .318 | .350 | .267 | .298 | .237 | .180 |
| NFCorpus [7] | 364.6 | .268 | .295 | .296 | **.308** | .295 | .288 | .284 | .151 |
| Vaswani | 51.3 | .447 | .306 | .414 | .458 | .321 | **.476** | .436 | .163 |
| WaPo | 713.0 | .364 | .451 | **.492** | .476 | .449 | .438 | .434 | .296 |
| Average | – | .358 | .353 | .374 | **.387** | .356 | .356 | .365 | .260 |

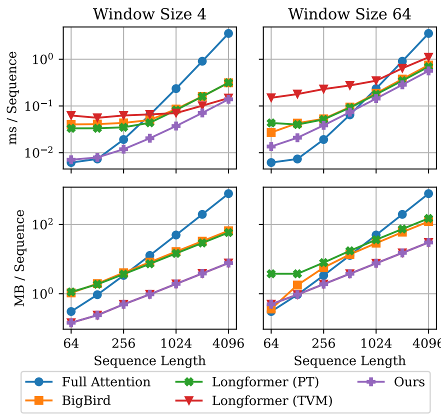
variants and the base variant of monoT5. We emphasize that our model only has about 24 million parameters, making it around four times smaller than the base variant of monoBERT, nine times smaller than the base variant of monoT5, and fourteen times smaller than monoBERT-large. It is slightly less effective than the monoT5 large variant, and the largest model, monoT5 3b, is the most effective.

In contrast, our model trained on documents is substantially less effective in out-of-domain retrieval. Across all corpora, it is 0.105 nDCG@10 less effective than our model trained on passages. However, it can take advantage of its longer context length on the corpora containing long documents. For example, on the ClueWeb corpora it is the most effective of all cross-encoder models and features competitive effectiveness on CORD-19, GOV, and GOV2.

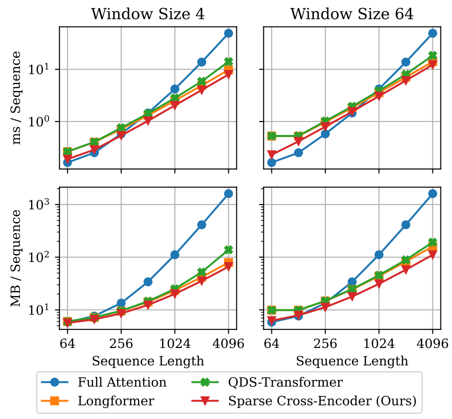
## 4.2 Efficiency Results

Finally, we study how the various attention patterns affect efficiency. We first compare our custom windowed matrix multiplication kernel with previous implementations. We then compare the efficiency of our proposed cross-encoder model to the reference cross-encoder, Longformer, and QDS-Transformer.

*Windowed Matrix Multiplication* Figure 2a compares our windowed matrix multiplication kernel with PyTorch's built-in matrix multiplication kernel (Full Attention), Longformer's TVM-based [11] implementation, and the two PyTorch-



(a) Matrix multiplication kernels.



(b) Cross-encoder models.

Figure 2: Comparison of windowed matrix multiplication kernels (a) and sparse cross-encoder models (b) in terms of efficiency. Time (ms / Document) and space (GB) efficiency are reported for window sizes $w \in \{4, 64\}$. All plots use a logarithmic scale with base 2 on the x-axis and base 10 on the y-axis.

based block-sparse implementations from BigBird and Longformer. All implementations are intended as drop-in replacements for matrix multiplication but use different ways to interface with CUDA and add varying levels of overhead.

We observe similar behavior as Beltagy et al. [3] for a window size of 64 tokens. Both block-sparse matrix implementations are time-efficient but sacrifice space efficiency. The opposite is true for the TVM-based kernel. For window size 4, the TVM implementation fairs better, achieving similar time efficiency as the PyTorch-based kernels for short sequence lengths and outperforming them at longer sequences. At the same time, it upholds its space efficiency. However, previous kernels are vastly slower compared to full matrix multiplication for shorter sequences. Our custom kernel achieves optimal space efficiency and is faster than all previous windowed matrix multiplication kernels for both window sizes. Compared to full matrix multiplication, our kernel is only slower for the edge case when the window size exceeds the sequence length.

*Cross-Encoders Models* Lastly, we compare the efficiency of our sparse cross-encoder model with a standard cross-encoder, the Longformer, and the QDS-Transformer. We use the default implementations from Huggingface [80] but omit BigBird, as it does not support task-specific global attention. Note that the QDS-Transformer is based on the Longformer and uses the same model architecture with a different global attention pattern.

Figure 2b gives a visual overview of the efficiency of the models for windows sizes 4 and 64. Table 3 reports the time and memory used per sequence for passages of length 164 and documents of length 4086. These lengths correspond to

Table 3: Time and space efficiency of cross-encoder models, including our sparse cross-encoder without our kernel and independent query contextualization. Relative differences to baseline models (underlined) are given in parentheses.

| **Unit** | **Full Att.** | **Longf.** | **QDS.** | **Sp. CE** | **Sp. CE** | **~~Kernel~~** | **~~Query~~** |
|---|---|---|---|---|---|---|---|
| $w =$ | $\infty$ | 64 | 64 | 64 | 4 | 4 | 4 |
| *Query length 10, Passage length 164* | | | | | | | |
| µs | <u>368</u> | 980 (+166%) | 995 (+170%) | 527 (+43%) | 364 (−1%) | 404 (+10%) | 403 (+10%) |
| MB | <u>9</u> | 15 (+67%) | 15 (+67%) | 9 (+0%) | 7 (−22%) | 8 (−11%) | 7 (−22%) |
| *Query length 10, Document length 4086* | | | | | | | |
| ms | 49 (+250%) | <u>14</u> | 18 (+29%) | 12 (−14%) | 8 (−43%) | 9 (−36%) | 8 (−43%) |
| MB | 1608 (+905%) | <u>160</u> | 192 (+20%) | 111 (−31%) | 66 (−59%) | 84 (−48%) | 66 (−59%) |

the average of the longest passage / document per top-100 ranking of a TREC Deep Learning query, i.e., the setup simulates re-ranking a batch of 100 sequences. For ablation analyses, Table 3 additionally reports the efficiency of our model without our custom kernel and independent query contextualization.

Figure 2b shows our model outperforms the other two sparse cross-encoders for time and space efficiency for both window sizes. The QDS-Transformer is the least efficient due to its additional global sentence attention. The Longformer lies between the QDS-Transformer and our sparse cross-encoder. The efficiency improvements can be attributed to two sources. The first is our improved windowed matrix multiplication kernel, and the second is our cross-attention mechanism. The Listformer uses a similar mechanism but extracts the matrices required for global attention in each transformer layer. Our sparse cross-encoder model splits the sub-sequences once and reuses the extracted matrices for all layers, avoiding repeating the expensive extraction and splitting step.

Table 3 underlines the efficiency improvements of our model. With a 64 token window size, our sparse cross-encoder is almost twice as fast and uses 40% less memory than the Longformer on passages. On documents, the difference is less pronounced but still substantial. Our model is 14% faster and uses 31% less memory. However, our sparse cross-encoder achieves the largest efficiency improvements when reducing the window size. Compared to the Longformer with a 64-token window size, our sparse cross-encoder with a 4-token window size is 63% faster and uses 53% less memory on passages. On documents, our model is 43% faster and uses 59% less memory. Despite the different window sizes, we deem this a fair comparison because the Longformer was previously not successfully used for re-ranking with smaller window sizes. It acts as the previous sparse cross-encoder efficiency standard.

Ablation tests show that our custom kernel and independent query contextualization both contribute to our model's improved efficiency. Using a Pytorch-based block-sparse windowed matrix multiplication kernel, our model is less time and space-efficient and loses between 9% and 11% percent of its time and space-efficiency improvements. Independent query contextualization only has a marginal effect on space efficiency and a noticeable effect on time efficiency only for passages. The query is generally not long enough compared to passages or documents to substantially impact efficiency in practice.

Comparing our sparse cross-encoder to the standard cross-encoder reveals that there is still room for improvement. Time and space efficiency on documents is orders of magnitude better, and our model uses 22% less memory on passages. But, regarding inference time, our model is on par with the standard cross-encoder for passages. The root cause is that the cross-attention incurs additional overhead. Each sub-sequence uses its own attention function. Multiple smaller attention functions are executed sequentially, while full attention uses a single large attention function for the entire sequence. Recent work on fused-attention kernels [31, 32, 48] has shown that moving the entire attention function to the GPU substantially improves efficiency. At the time of writing, fused-attention kernels do not support asymmetric attention patterns. We leave investigating their applicability to our model to future work.

## 5 Conclusion

We have investigated the impact of sparse attention on the re-ranking effectiveness of cross-encoders by combining windowed self-attention and token-specific cross-attention to analyze (1) decreasing context sizes for document tokens and (2) deactivating attention from the query to the [CLS] and document tokens.

In passage and document re-ranking experiments, we find a window size down to four tokens to be as effective as larger window sizes or full attention (significantly equivalent effectiveness within $\pm 0.02$ nDCG@10 compared to previous cross-encoders), and we find that deactivating attention from the query to the [CLS] and document tokens does not impact effectiveness. At the same time, combining the sparsification techniques substantially improves efficiency of passage and document re-ranking. For these efficiency improvements, our custom CUDA kernel and asymmetric cross-attention play substantial roles but the largest gains are achieved using small window sizes.

Sparse attention thus is a viable option for decreasing computational effort without substantially affecting effectiveness. To further increase efficiency, integrating asymmetric cross-attention and windowed self-attention into newly developed fused attention kernels [31, 32, 48] seems to be a promising direction. The flexibility and efficiency of our custom attention pattern also allow for further research into the direction of listwise re-ranking by passing multiple documents to the cross-encoder at once.

## Acknowledgments

This work has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101070014 (OpenWebSearch.EU, https://doi.org/10.3030/101070014).

## References

[1] Ainslie, J., Ontañón, S., Alberti, C., Cvicek, V., Fisher, Z., Pham, P., Ravula, A., Sanghai, S., Wang, Q., Yang, L.: ETC: Encoding Long and Structured Inputs in Transformers. In: Proceedings of EMNLP 2020, pp. 268–284 (2020), https://doi.org/10.18653/v1/2020.emnlp-main.19

[2] Arapakis, I., Bai, X., Cambazoglu, B.B.: Impact of Response Latency on User Behavior in Web Search. In: Proceedings of SIGIR 2014, pp. 103–112 (2014), https://doi.org/10.1145/2600428.2609627

[3] Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The Long-Document Transformer. arXiv (2020), https://doi.org/10.48550/arXiv.2004.05150

[4] Bevendorff, J., Stein, B., Hagen, M., Potthast, M.: Elastic ChatNoir: Search Engine for the ClueWeb and the Common Crawl. In: Proceedings of ECIR 2018, pp. 820–824 (2018), https://doi.org/10.1007/978-3-319-76941-7_83

[5] Bondarenko, A., Fröbe, M., Kiesel, J., Syed, S., Gurcke, T., Beloucif, M., Panchenko, A., Biemann, C., Stein, B., Wachsmuth, H., Potthast, M., Hagen, M.: Overview of Touché 2022: Argument Retrieval. In: Proceedings of CLEF 2022, pp. 311–336 (2022), https://doi.org/10.1007/978-3-031-13643-6_21

[6] Bondarenko, A., Gienapp, L., Fröbe, M., Beloucif, M., Ajjour, Y., Panchenko, A., Biemann, C., Stein, B., Wachsmuth, H., Potthast, M., Hagen, M.: Overview of Touché 2021: Argument Retrieval. In: Proceedings of CLEF 2021, pp. 450–467 (2021), https://doi.org/10.1007/978-3-030-85251-1_28

[7] Boteva, V., Ghalandari, D.G., Sokolov, A., Riezler, S.: A Full-Text Learning to Rank Dataset for Medical Information Retrieval. In: Proceedings of ECIR 2016, pp. 716–722 (2016)

[8] Burges, C.J.C.: From RankNet to LambdaRank to LambdaMART: An Overview. Technical Report, Microsoft Research (2010)

[9] Büttcher, S., Clarke, C.L.A., Soboroff, I.: The TREC 2006 Terabyte Track. In: Proceedings of TREC 2006, NIST Special Publication, vol. 500-272 (2006)

[10] Chen, S., Wong, S., Chen, L., Tian, Y.: Extending Context Window of Large Language Models via Positional Interpolation. arXiv (2023), https://doi.org/10.48550/arXiv.2306.15595

[11] Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E., Cowan, M., Shen, H., Wang, L., Hu, Y., Ceze, L., Guestrin, C., Krishnamurthy, A.: TVM: An automated end-to-end optimizing compiler for deep learning. In: Proceedings of OSDI 2018, pp. 579–594 (2018)

[12] Child, R., Gray, S., Radford, A., Sutskever, I.: Generating Long Sequences with Sparse Transformers. arXiv (2019), https://doi.org/10.48550/arXiv.1904.10509

[13] Clarke, C.L.A., Craswell, N., Soboroff, I.: Overview of the TREC 2004 Terabyte Track. In: Proceedings of TREC 2004, NIST Special Publication, vol. 500-261 (2004)

[14] Clarke, C.L.A., Craswell, N., Soboroff, I.: Overview of the TREC 2009 Web Track. In: Voorhees, E.M., Buckland, L.P. (eds.) Proceedings of TREC 2009, NIST Special Publication, vol. 500-278 (2009)

[15] Clarke, C.L.A., Craswell, N., Soboroff, I., Cormack, G.V.: Overview of the TREC 2010 Web Track. In: Voorhees, E.M., Buckland, L.P. (eds.) Proceedings of TREC 2010, NIST Special Publication, vol. 500-294 (2010)

[16] Clarke, C.L.A., Craswell, N., Soboroff, I., Voorhees, E.M.: Overview of the TREC 2011 Web Track. In: Voorhees, E.M., Buckland, L.P. (eds.) Proceedings of TREC 2011, NIST Special Publication, vol. 500-296 (2011)

[17] Clarke, C.L.A., Craswell, N., Voorhees, E.M.: Overview of the TREC 2012 Web Track. In: Voorhees, E.M., Buckland, L.P. (eds.) Proceedings of TREC 2012, NIST Special Publication, vol. 500-298 (2012)
[18] Clarke, C.L.A., Scholer, F., Soboroff, I.: The TREC 2005 Terabyte Track. In: Proceedings of TREC 2005, NIST Special Publication, vol. 500-266 (2005)
[19] Cleverdon, C.: The Cranfield Tests on Index Language Devices. In: ASLIB Proceedings, pp. 173–192, MCB UP Ltd. (Reprinted in Readings in Information Retrieval, Karen Sparck-Jones and Peter Willett, editors, Morgan Kaufmann, 1997) (1967)
[20] Cleverdon, C.W.: The Significance of the Cranfield Tests on Index Languages. In: Bookstein, A., Chiaramella, Y., Salton, G., Raghavan, V.V. (eds.) Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Chicago, Illinois, USA, October 13-16, 1991 (Special Issue of the SIGIR Forum), pp. 3–12 (1991)
[21] Collins-Thompson, K., Bennett, P.N., Diaz, F., Clarke, C., Voorhees, E.M.: TREC 2013 web track overview. In: Proceedings of TREC 2013, NIST Special Publication, vol. 500-302 (2013)
[22] Collins-Thompson, K., Macdonald, C., Bennett, P.N., Diaz, F., Voorhees, E.M.: TREC 2014 web track overview. In: Proceedings of TREC 2014, NIST Special Publication, vol. 500-308 (2014)
[23] Craswell, N., Hawking, D.: Overview of the TREC 2002 Web Track. In: Proceedings of TREC 2002, NIST Special Publication, vol. 500-251 (2002)
[24] Craswell, N., Hawking, D.: Overview of the TREC 2004 Web Track. In: Proceedings of TREC 2004, NIST Special Publication, vol. 500-261 (2004)
[25] Craswell, N., Hawking, D., Wilkinson, R., Wu, M.: Overview of the TREC 2003 Web Track. In: Proceedings of TREC 2003, NIST Special Publication, vol. 500-255, pp. 78–92 (2003)
[26] Craswell, N., Mitra, B., Yilmaz, E., Campos, D.: Overview of the TREC 2020 Deep Learning Track. In: Proceedings TREC 2020, NIST Special Publication, vol. 1266 (2020)
[27] Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Lin, J.: Overview of the TREC 2021 Deep Learning Track. In: Proceedings TREC 2021, NIST Special Publication, vol. 500-335 (2021)
[28] Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Voorhees, E.M.: Overview of the TREC 2019 Deep Learning Track. In: Proceedings TREC 2019, NIST Special Publication, vol. 500-331 (2019)
[29] Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Voorhess, J.L.E.M., Soboroff, I.: Overview of the TREC 2022 Deep Learning Track. In: Proceedings TREC 2022, NIST Special Publication, vol. 500-338 (2022)
[30] Dai, Z., Callan, J.: Deeper Text Understanding for IR with Contextual Neural Language Modeling. In: Proceedings of SIGIR 2019, pp. 985–988 (2019), https://doi.org/10.1145/3331184.3331303
[31] Dao, T.: FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. arXiv (2023), https://doi.org/10.48550/arXiv.2307.08691
[32] Dao, T., Fu, D.Y., Ermon, S., Rudra, A., Ré, C.: FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In: Proceedings of NeurIPS 2022, pp. 16344–16359 (2022)
[33] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of NAACL-HLT 2019, pp. 4171–4186 (2019), https://doi.org/10.18653/v1/N19-1423

[34] Falcon, W., The PyTorch Lightning team: PyTorch Lightning (2023), https://doi.org/10.5281/zenodo.7859091
[35] Feng, C., Wang, X., Zhang, Y., Zhao, C., Song, M.: CASwin Transformer: A Hierarchical Cross Attention Transformer for Depth Completion. In: Proceedings of ITSC 2022, pp. 2836–2841 (2022), https://doi.org/10.1109/ITSC55140.2022.9922273
[36] Fröbe, M., Reimer, J.H., MacAvaney, S., Deckers, N., Reich, S., Bevendorff, J., Stein, B., Hagen, M., Potthast, M.: The Information Retrieval Experiment Platform. In: Proceedings of SIGIR 2023, pp. 2826–2836 (2023), https://doi.org/10.1145/3539618.3591888
[37] Hashemi, H., Aliannejadi, M., Zamani, H., Croft, W.B.: ANTIQUE: A Non-factoid Question Answering Benchmark. In: Proceedings of ECIR 2020, pp. 166–173 (2020)
[38] Hersh, W.R., Bhupatiraju, R.T., Ross, L., Cohen, A.M., Kraemer, D., Johnson, P.: TREC 2004 Genomics Track Overview. In: Proceedings of TREC 2004, NIST Special Publication, vol. 500-261 (2004)
[39] Hersh, W.R., Cohen, A.M., Yang, J., Bhupatiraju, R.T., Roberts, P.M., Hearst, M.A.: TREC 2005 Genomics Track Overview. In: Proceedings of TREC 2005, NIST Special Publication, vol. 500-266 (2005)
[40] Hofstätter, S., Althammer, S., Schröder, M., Sertkan, M., Hanbury, A.: Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. arXiv (2021), https://doi.org/10.48550/arXiv.2010.02666
[41] Hofstätter, S., Lin, S.C., Yang, J.H., Lin, J., Hanbury, A.: Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. In: Proceedings of SIGIR 2021, pp. 113–122 (2021), https://doi.org/10.1145/3404835.3462891
[42] Hofstätter, S., Zlabinger, M., Hanbury, A.: Interpretable & Time-Budget-Constrained Contextualization for Re-Ranking. In: Proceedings of ECAI 2020, pp. 513–520 (2020), https://doi.org/10.3233/FAIA200133
[43] Ilinykh, N., Dobnik, S.: Attention as Grounding: Exploring Textual and Cross-Modal Attention on Entities and Relations in Language-and-Vision Transformer. In: Findings of ACL 2022, pp. 4062–4073 (2022), https://doi.org/10.18653/v1/2022.findings-acl.320
[44] Jiang, J.Y., Xiong, C., Lee, C.J., Wang, W.: Long Document Ranking with Query-Directed Sparse Transformer. In: Findings of EMNLP 2020, pp. 4594–4605 (2020), https://doi.org/10.18653/v1/2020.findings-emnlp.412
[45] Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.t.: Dense Passage Retrieval for Open-Domain Question Answering. In: Proceedings of EMNLP 2020, pp. 6769–6781 (2020), https://doi.org/10.18653/v1/2020.emnlp-main.550
[46] Khattab, O., Zaharia, M.: ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In: Proceedings of SIGIR 2020, pp. 39–48 (2020), https://doi.org/10.1145/3397271.3401075
[47] Lauzon, C., Caffo, B.: Easy Multiplicity Control in Equivalence Testing Using Two One-sided Tests. The American Statistician **63**, 147–154 (2009), ISSN 0003-1305, https://doi.org/10.1198/tast.2009.0029
[48] Lefaudeux, B., Massa, F., Liskovich, D., Xiong, W., Caggiano, V., Naren, S., Xu, M., Hu, J., Tintore, M., Zhang, S., Labatut, P., Haziza, D.: xFormers: A modular and hackable Transformer modelling library. https://github.com/facebookresearch/xformers (2022)

[49] Lin, J., Ma, X., Lin, S.C., Yang, J.H., Pradeep, R., Nogueira, R.: Pyserini: An Easy-to-Use Python Toolkit to Support Replicable IR Research with Sparse and Dense Representations. In: Proceedings of SIGIR 2021, pp. 2356–2362 (2021), https://doi.org/10.1145/3404835.3463238

[50] Lin, S.C., Yang, J.H., Lin, J.: In-Batch Negatives for Knowledge Distillation with Tightly-Coupled Teachers for Dense Retrieval. In: Proceedings of RepL4NLP 2021, pp. 163–173 (2021), https://doi.org/10.18653/v1/2021.repl4nlp-1.17

[51] Loshchilov, I., Hutter, F.: Decoupled Weight Decay Regularization. In: Proceedings of ICLR 2019 (2019)

[52] MacAvaney, S., Nardini, F.M., Perego, R., Tonellotto, N., Goharian, N., Frieder, O.: Efficient Document Re-Ranking for Transformers by Precomputing Term Representations. In: Proceedings of SIGIR 2020, pp. 49–58 (2020), https://doi.org/10.1145/3397271.3401093

[53] MacAvaney, S., Yates, A., Feldman, S., Downey, D., Cohan, A., Goharian, N.: Simplified Data Wrangling with ir_datasets. In: Proceedings of SIGIR 2021, pp. 2429–2436 (2021), https://doi.org/10.1145/3404835.3463254

[54] Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., Deng, L.: MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In: Proceedings of COCO@NeurIPS 2016 (2016)

[55] Nogueira, R., Cho, K.: Passage Re-ranking with BERT. arXiv (2020), https://doi.org/10.48550/arXiv.1901.04085

[56] Nogueira, R., Jiang, Z., Pradeep, R., Lin, J.: Document Ranking with a Pretrained Sequence-to-Sequence Model. In: Findings of EMNLP 2020, pp. 708–718 (2020), https://doi.org/10.18653/v1/2020.findings-emnlp.63

[57] Nogueira, R., Yang, W., Cho, K., Lin, J.: Multi-Stage Document Ranking with BERT. arXiv (2019), https://doi.org/10.48550/arXiv.1910.14424

[58] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Proceedings of NeurIPS 2019, vol. 32 (2019)

[59] Petit, O., Thome, N., Rambour, C., Themyr, L., Collins, T., Soler, L.: U-Net Transformer: Self and Cross Attention for Medical Image Segmentation. In: Proceedings of MLMI@MICCAI 2021, pp. 267–276 (2021), https://doi.org/10.1007/978-3-030-87589-3_28

[60] Potthast, M., Hagen, M., Stein, B., Graßegger, J., Michel, M., Tippmann, M., Welsch, C.: ChatNoir: A search engine for the ClueWeb09 corpus. In: Proceedings of SIGIR 2012, p. 1004 (2012), https://doi.org/10.1145/2348283.2348429

[61] Qu, Y., Ding, Y., Liu, J., Liu, K., Ren, R., Zhao, W.X., Dong, D., Wu, H., Wang, H.: RocketQA: An Optimized Training Approach to Dense Passage Retrieval for Open-Domain Question Answering. In: Proceedings of NAACL-HLT 2021, pp. 5835–5847 (2021), https://doi.org/10.18653/v1/2021.naacl-main.466

[62] Reimers, N., Gurevych, I.: Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: Proceedings of EMNLP-IJCNLP 2019, pp. 3980–3990 (2019), https://doi.org/10.18653/v1/D19-1410

[63] Roberts, K., Demner-Fushman, D., Voorhees, E.M., Hersh, W.R., Bedrick, S., Lazar, A.J.: Overview of the TREC 2018 Precision Medicine Track. In: Proceedings of TREC 2018, NIST Special Publication, vol. 500-331 (2018)

[64] Roberts, K., Demner-Fushman, D., Voorhees, E.M., Hersh, W.R., Bedrick, S., Lazar, A.J., Pant, S.: Overview of the TREC 2017 Precision Medicine Track. In: Proceedings of TREC 2017, NIST Special Publication, vol. 500-324 (2017)

[65] Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M., Gatford, M.: Okapi at TREC-3. In: Proceedings of TREC 1994, vol. 500–225, pp. 109–126 (1994)

[66] Rosa, G., Bonifacio, L., Jeronymo, V., Abonizio, H., Fadaee, M., Lotufo, R., Nogueira, R.: In Defense of Cross-Encoders for Zero-Shot Retrieval. arXiv (2022), https://doi.org/10.48550/arXiv.2212.06121

[67] Santhanam, K., Khattab, O., Saad-Falcon, J., Potts, C., Zaharia, M.: ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. In: Proceedings of NAACL-HLT 2022, pp. 3715–3734 (2022), https://doi.org/10.18653/v1/2022.naacl-main.272

[68] Scells, H., Zhuang, S., Zuccon, G.: Reduce, Reuse, Recycle: Green Information Retrieval Research. In: Proceedings of SIGIR 2022, pp. 2825–2837 (2022), https://doi.org/10.1145/3477495.3531766

[69] Schuirmann, D.J.: A Comparison of the Two One-Sided Tests Procedure and the Power Approach for Assessing the Equivalence of Average Bioavailability. Journal of Pharmacokinetics and Biopharmaceutics **15**(6), 657–680 (Dec 1987), https://doi.org/10.1007/BF01068419

[70] Sekulić, I., Soleimani, A., Aliannejadi, M., Crestani, F.: Longformer for MS MARCO Document Re-ranking Task. In: Proceedings of TREC 2020, NIST Special Publication, vol. 1266 (2020)

[71] Sui, X., Li, S., Geng, X., Wu, Y., Xu, X., Liu, Y., Goh, R., Zhu, H.: CRAFT: Cross-Attentional Flow Transformer for Robust Optical Flow. In: Proceedings of CVPR 2022, pp. 17581–17590 (2022), https://doi.org/10.1109/CVPR52688.2022.01708

[72] Tay, Y., Dehghani, M., Bahri, D., Metzler, D.: Efficient Transformers: A Survey. ACM Computing Surveys **55**, 109:1–109:28 (2023), https://doi.org/10.1145/3530811

[73] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is All you Need. In: Proceedings of NeurIPS 2017, pp. 5998–6008 (2017)

[74] Voorhees, E.M.: NIST TREC Disks 4 and 5: Retrieval Test Collections Document Set (1996)

[75] Voorhees, E.M.: Overview of the TREC 2004 Robust Track. In: Proceedings of TREC 2004, NIST Special Publication (2004)

[76] Voorhees, E.M., Harman, D.: Overview of the Seventh Text Retrieval Conference (TREC-7). In: Proceedings of TREC 1998, NIST Special Publication (1998)

[77] Voorhees, E.M., Harman, D.: Overview of the Eight Text Retrieval Conference (TREC-8). In: Proceedings of TREC 1999, NIST Special Publication (1999)

[78] Wang, L.L., Lo, K., Chandrasekhar, Y., Reas, R., Yang, J., Eide, D., Funk, K., Kinney, R., Liu, Z., Merrill, W., Mooney, P., Murdick, D.A., Rishi, D., Sheehan, J., Shen, Z., Stilson, B., Wade, A.D., Wang, K., Wilhelm, C., Xie, B., Raymond, D., Weld, D.S., Etzioni, O., Kohlmeier, S.: CORD-19: The Covid-19 Open Research Dataset. arXiv (2020), https://doi.org/10.48550/arXiv.2004.10706

[79] Wang, W., Bao, H., Huang, S., Dong, L., Wei, F.: MiniLMv2: Multi-Head Self-Attention Relation Distillation for Compressing Pretrained Transformers. In: Findings of ACL-IJCNLP 2021, pp. 2140–2151 (2021), https://doi.org/10.18653/v1/2021.findings-acl.188

[80] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv (2020), https://doi.org/10.48550/arXiv.1910.03771
[81] Xiong, L., Xiong, C., Li, Y., Tang, K.F., Liu, J., Bennett, P.N., Ahmed, J., Overwijk, A.: Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In: Proceedings of ICLR 2021 (2021)
[82] Yan, M., Li, C., Wu, C., Bi, B., Wang, W., Xia, J., Si, L.: IDST at TREC 2019 Deep Learning Track: Deep Cascade Ranking with Generation-based Document Expansion and Pre-trained Language Modeling. In: Proceedings of TREC 2019, NIST Special Publication, vol. 1250 (2019)
[83] Zaheer, M., Guruganesh, G., Dubey, K.A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., Ahmed, A.: Big Bird: Transformers for Longer Sequences. In: Proceedings of NeurIPS 2020, pp. 17283–17297 (2020)
[84] Zhang, Y., Long, D., Xu, G., Xie, P.: HLATR: Enhance Multi-stage Text Retrieval with Hybrid List Aware Transformer Reranking. arXiv (2022), https://doi.org/10.48550/arXiv.2205.10569