Intra-Document Cascading: Learning to Select Passages for Neural Document Ranking

Anonymous Author(s)

ABSTRACT

An emerging recipe for achieving state-of-the-art effectiveness in neural document re-ranking involves utilizing large pre-trained language models—*e.g.*, BERT—to evaluate all individual passages in the document and then aggregating the outputs by pooling or additional Transformer layers. A major drawback of this approach is high query latency due to the cost of evaluating every passage in the document with BERT. To make matters worse, this high inference cost and latency varies based on the length of the document, with longer documents requiring more time and computation. To address this challenge, we adopt an *intra-document cascading* strategy, which prunes passages of a candidate document using a less expensive model, called ESM, before running a scoring model that is more expensive and effective, called ETM. We found it best to train ESM via knowledge distillation from the ETM model (e.g., BERT). This pruning allows us to only run the ETM model on a smaller set of passages whose size does not vary by document length. Our experiments on the MS MARCO and TREC Deep Learning Track benchmarks suggest that the proposed Intra-Document Cascaded Ranking Model (IDCM) leads to over 400% lower query latency by providing essentially the same effectiveness as the state-of-the-art BERT-based document ranking models.

ACM Reference Format:
Anonymous Author(s). 2021. Intra-Document Cascading: Learning to Select Passages for Neural Document Ranking. In *SIGIR '21.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Ranking documents in response to a query is a core problem in information retrieval (IR). Many systems incorporate ranking as a core component, such as Web search engines and news search systems. Other systems build on document ranking, e.g., an agent capable of conversation and disambiguation may still have document ranking as a component [40]. Therefore, retrieval model improvements are likely to lead to "lifting all boats".

One difficulty in document ranking is that documents can vary significantly in length. In traditional IR, variation in length can be explained by (1) the verbosity hypothesis, that the author used more words to explain the topic, or (2) the scope hypothesis, that the author covered multiple topics [27]. In practice, each hypothesis is a partial explanation for document length, and retrieval models typically apply document length normalization. In other words, a document with some irrelevant parts can still be considered relevant overall, because having somewhat broader scope than the current query does not rule out a document from being a useful search result.

One way to deal with documents of varying length with some irrelevant parts is to use passage-level evidence for document ranking. Early studies [4, 16] found that a passage of text need not be defined based on document structure, such as paragraphs or sentences. A good approach was to divide the content into fixed-size windows of 150 words or more, compare the query to all passages, then score the document based on the score of its highest-scoring passage. This is consistent with the scope hypothesis, that we should focus on finding some relevant content, without penalizing the document for having some irrelevant content.

In neural IR, it is possible to significantly outperform classic retrieval systems [7, 19]. This is often done by taking multiple fixed-size windows of text, applying a deep neural network to score each passage, and scoring the document based on the highest-scoring passages. This is similar to the classic IR approaches in [4, 16], but works much better as the per-passage models are more effective.

However, the problem with the neural approaches is the cost of applying the per-passage model. Applying neural net inference for every passage in every document being ranked requires significant computation and leads to higher query latency. This limits the impact of the new approach, since if the cost of inference is too high, it cannot be used in large-scale production systems. To avoid this problem, some models retrieve documents solely based on their first passage [7], however, this is a sub-optimal solution.

In this work, we address this issue by proposing an Intra-Document Cascading Model (IDCM) that employs an in-document cascading mechanism with a fast selection module and a slower effective scoring module.[1] This simultaneously provides lower query latency and state-of-the-art ranking effectiveness. We evaluate our model on two document ranking query sets: (i) TREC DL 2019 [7], (ii) MSMARCO [1]. We study how to train the IDCM architecture in multiple stages to control the collaboration of the cascading sub-modules, and investigate:

RQ1 Can IDCM achieve comparable effectiveness to the full BERT-based ranker at lower computation cost and query latency?

Among the different variants we explore in this work, we found that training the selection module using *knowledge distillation* based on passage-level labels derived from the BERT score in conjunction with an intra-document ranking loss achieves the best overall re-ranking quality. Under this setting, the selection module is trained to approximate the passage ranking that would have been produced if ordered by their corresponding BERT-scores, to filter out non-relevant parts of the document.

[1] In our experiments, we used DistilBERT [28], an efficient variant of BERT as the "slow effective scoring module". Throughout this paper, we refer to it as the BERT model.

SIGIR '21, Online, 2021,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

An important hyperparameter in this setting is the number of passages per document that survives the selection stage for the subsequent BERT-based evaluation. Here we study:

RQ2 How is the effectiveness-efficiency trade-off influenced by the number of passages k that the less expensive model selects from the document?

In our baseline setting, the BERT model inspects up to the first k passages. We observe superior performance with $k = 40$ compared to smaller values of k. However, the proposed IDCM framework achieves roughly the same ranking effectiveness by applying the BERT model to only the top 4 passages pre-selected by the less-expensive preceding model. Consequently, this result comes at a much lower query latency.

The application of BERT to multiple in-document passages has the undesirable property of introducing large variance in query response time depending on the length of candidate documents. Under the IDCM settings, this variance is largely reduced because the expensive BERT model is applied to k top passages for every document, unless the document is so short that it has less than k passages. This leads us to our next research question:

RQ3 How does IDCM compare to the baseline with respect to variance in query latency?

We observe that our baseline setup of using BERT over all passages has a very high standard deviation and long tail *w.r.t.* query latency, whereas IDCM has much more predictable query latency centered around the mean. This is partly because our passage selection module is fast enough to triage up to 40 candidate passages in the same time as the BERT model takes to evaluate a single passage. Therefore, while the contribution of the selection stage to query latency is still a function of document length, the variance is largely reduced.

Finally, we study the how the passage selection under IDCM compares to the highest-scoring passages by the BERT model.

RQ4 How often does the passage selection under IDCM recall the same passages as those scored highly by the BERT model?

Overall, we observe that the selection module recalls 60-85% of the top BERT-scored passages depending on the value of k. We also find that passages closer to the beginning of the document are more likely to be selected by both models, although there is a long tail of passage selection from lower positions within the document. Interestingly, we also observe that the selection module achieves a better recall in case of relevant documents, which may be explained by the relevant passages being easier to distinguish from the nonrelevant passages in a relevant document, compared to the same in case of a nonrelevant document.

To summarize, this paper proposes a cascaded architecture and training regime to address the high and variable query latency associated with applying BERT to evaluate all in-document passages for the document ranking task. We demonstrate that employing our approach leads to lower mean and variance for query latency, and enables broader scope of application for BERT-based ranking models in real-world retrieval systems.

- We propose IDCM, an intra-document cascade ranking model, including a training workflow using knowledge distillation.
- We evaluate our approach on TREC-DL'19 and MSMARCO DEV; and show that IDCM achieves similar effectiveness on average at four times lesser latency compared to the BERT-only ranking model.
- We perform extensive ablation studies to validate our multi-stage training approach and the benefits of knowledge distillation for optimizing the selection module of IDCM.

To improve the reproducibility of our work, we open-source our implementation and release the trained models at https://github.com/anonymousURL.

2 RELATED WORK

Since demonstrating impressive performance on passage ranking [23], several successful efforts are underway to employ BERT [11] and other Transformer [31] based models to the document ranking task [7, 14]. However, the quadratic memory complexity of the self-attention layer with respect to the input sequence length poses a unique challenge in the way of scaling these models to documents that may contain thousands of terms. While some efforts are underway to directly address the scalability of the self-attention layer in these ranking models [22], a majority of applications of Transformer-based architectures to document ranking involves segmenting the document text into smaller chunks of text that can be processed efficiently [14, 18, 37].

The idea of using passage-level evidence for document ranking is not unique to neural methods. Several classical probabilistic [4] and language model [2, 20] based retrieval methods—as well as machine learning based approaches [29]—incorporate passage-based relevance signals for document ranking. An underlying assumption across many of these approaches is that all passages from the document are inspected by the model. However, as our models become more computation and memory intensive, it quickly becomes expensive and even infeasible to evaluate every passage from the document. This is exactly where text selection using light-weight approaches becomes necessary so that the more complex model only has to inspect the most important parts of the document, which is the main motivation for this work.

One could draw parallels between our approach of using cheaper models to detect interesting regions of the document to the highly influential work of Viola and Jones [32] in computer vision for fast object detection using cascades of models of increasing complexity. Similarly, cascaded approaches have also been employed extensively [5, 12, 24, 33] in IR to progressively rank-and-prune the set of candidate documents, from the full collection all the way to the final ten or so results presented to the user. Unlike these approaches that employ a cascade of models to prune the candidate set of documents, we use the cascaded approach to prune the set of regions within the document that needs to be inspected by the expensive and effective ranking model (i.e., intra-document cascading).

In a cascaded setting, typically the models are trained progressively, starting with the simplest model that is exposed to the largest number of candidates, and then subsequent models are trained using a data distribution that reflects the candidates that survive the earlier stages of pruning. This sequential training strategy is sometimes referred to as telescoping [21]. Joint optimization of the

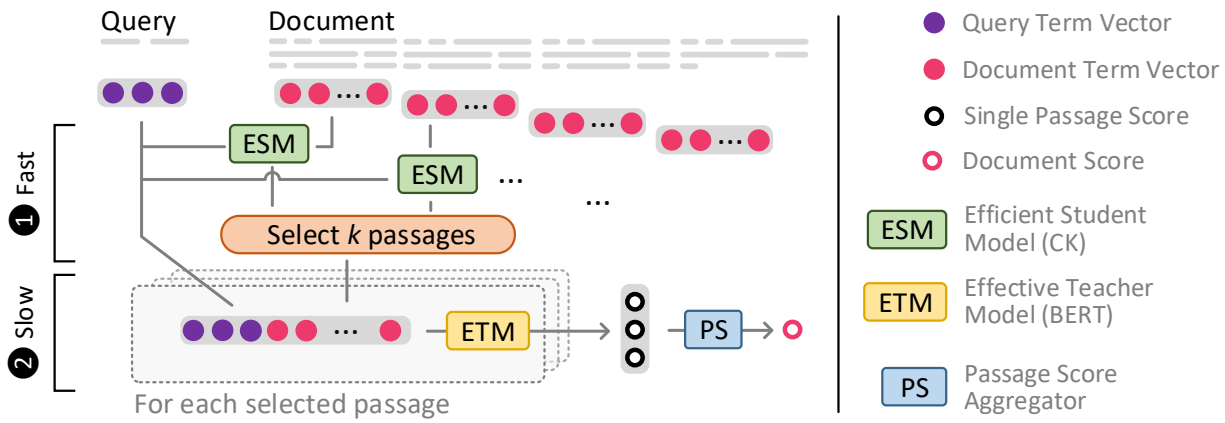


Figure 1: The IDCM architecture that consists of two cascading stages: ❶ To allow for long-document input, the first stage is a lightweight and fast selection model. ❷ Only the top k passages from the selection model are scored with a costly BERT-based scoring module to form the final document score.

different rankers within a multi-stage cascaded search architecture has also been explored [12].

In our work, we adopt a different training strategy. We begin by training the costly ranking model and then apply knowledge distillation to train the model for the preceding selection stage. Our approach is motivated by the strong empirical performance observed in recent work exploring knowledge distillation from larger BERT to more effective models [13, 15, 28, 30].

3 THE INTRA-DOCUMENT CASCADE MODEL

Neural ranking models have resulted in significant improvements in a wide variety of information retrieval tasks. However, there exists an efficiency-effectiveness trade-off in these models. In other words, state-of-the-art neural ranking models mostly suffer from high query latency and high GPU memory requirements. A popular solution to address the GPU memory issue is to divide documents into multiple passages and compute the retrieval scores for each passage [25, 37, 39]. For instance, Dai and Callan [8] compare considering only the first passage of the document with scoring every passage in the document and use the highest passage score as the document score. We believe that the existing approaches lead to either sub-optimal ranking or high computational cost.

Inspired by previous work on passage-level evidences for document retrieval [4, 20], in this section we propose an alternative efficient and effective solution by introducing the Intra-Document Cascade Model named **IDCM**. In the following, we first introduce the IDCM architecture and optimization, and then describe its implementation details.

3.1 The IDCM Architecture

IDCM is designed based on a cascade architecture within the documents. For each query-document pair, the idea is to select a few passages from the document using an efficient model and then produce the retrieval score for the document by exploiting a more expensive and effective model on the selected passages. The high-level architecture of IDCM is presented in Figure 1.

IDCM takes a query q and a document d as input. It first divides the document into multiple units of partially overlapping windows of size w with an overlapping factor of o, where $o < w$. This results in $\lceil dl/w \rceil$ passages as follows:

$$P = [(d_{1-o:w+o})(d_{w-o:2w+o})(d_{2w-o:3w+o}); ...] \tag{1}$$

Each passages contains $w + 2o$ tokens with exactly $2o + 1$ tokens in common with their previous and next passages respectively. The first and the last passages are padded. A key practical impact of padding the windows (as opposed to padding the document) is the possibility to compact batched tensor representations and skip padding-only windows entirely for batches that contain documents of different lengths.

IDCM uses an efficient model to score each passage $p \in P$ with respect to the query. This model is called **ESM**. To cascade the scoring decision and discard a large number of passages, IDCM selects the top k passages with the highest scores produced by the ESM model, as follows:

$$\hat{P} = \underset{\hat{P} \subseteq P, |\hat{P}|=k}{\arg\max} \sum_{p \in \hat{P}} \text{ESM}(q, p) \tag{2}$$

It is important to keep the size of $\hat{P}$ as small as possible to use the efficiency advantages brought by the cascading approach.

The selected passages are then scored by a more expensive and effective ranking model, called **ETM**:

$$S_{\text{ETM}} = \text{ETM}(q, p) \,\big|\, p \in \hat{P} \tag{3}$$

Note that the token embedding parameters are shared between the ESM and ETM models. We compute the document relevance score using a weighted linear interpolation. In other words, we feed the top l sorted ETM scores to a fully-connected layer to produce

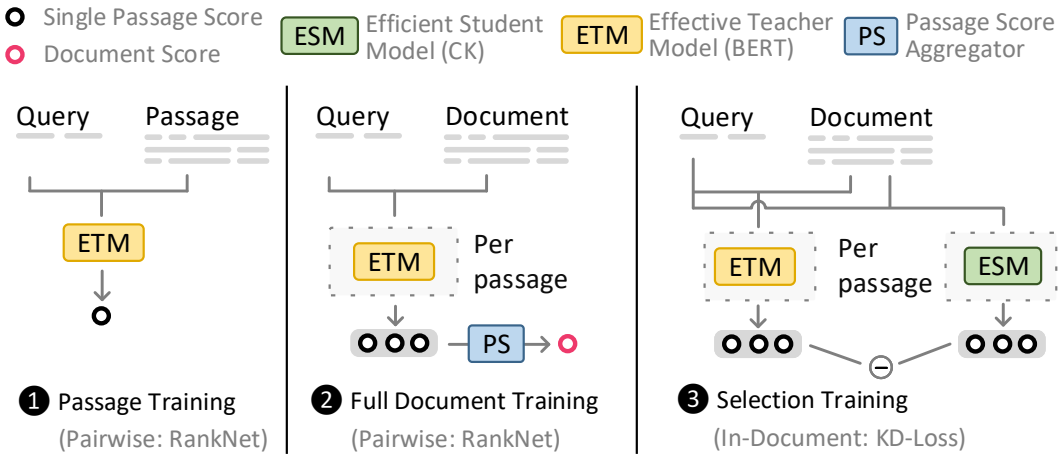


Figure 2: The staged training workflow of IDCM: ❶ Training the ETM (BERT) passage module ❷ Training the full model on a document collection without selection (all available passages of a document are scored with ETM). ❸ The ESM (CK) selection module is now trained via knowledge distillation using the ETM (BERT) scores as labels.

the document relevance score as follows:

$$\text{IDCM}(q, d) = \text{top}_l(S_{\text{ETM}}) * W_{PS} \tag{4}$$

where W_{PS} is a $l \times 1$ weight matrix for linear interpolation of the passage scores.

3.2 The IDCM Optimization

The IDCM framework consists of multiple non-differentiable operations (e.g., passage selection) that makes it difficult to use gradient descent-based methods for end-to-end optimization. Therefore, we split the IDCM optimization to three steps with different objectives, as shown in Figure 2. These three steps include: (1) optimizing the ETM model for passage ranking, (2) extending the ETM optimization to full document ranking, and (3) optimizing the ESM model for passage selection using knowledge distillation. Each step completes with early stopping based on the performance on a held-out validation set and the best model checkpoint is used as the initialization for the following step(s). The first step involves training the passage ranking model.

Step I: Optimizing ETM for Passage Ranking. The first training step is to train the ETM model on a passage collection. To this aim, we adopt the pairwise ranking loss function used in RankNet [3]. In more detail, for a given pair of negative and positive passages p^- and p^+ for the query q in the training set, we use a binary cross-entropy loss function for pairwise passage ranking optimization as follows:

$$\mathcal{L}_{\text{Pas.}}(q, p^+, p^-) = -\log \sigma(\text{ETM}(q, p^+) - \text{ETM}(q, p^-)) \tag{5}$$

where $\sigma(\cdot)$ is the sigmoid function.

This step prepares the ETM model for the document retrieval task. Such pre-training has been successfully employed in recent models, such as PARADE [18]. The parallel MSMARCO passage and document collections make this pre-training possible, albeit it remains optional if passage relevance is not available, as the BERT module is also trained in the next step.

Step II: Extending the ETM Optimization to Full-Document Ranking. Optimizing the model for passage ranking is not sufficient for document retrieval, mainly because of the following two reasons: First, the passage aggregation parameters (i.e., W_{PS}) need to be optimized; Second, the passage and document collections may exhibit different assumptions on what constitutes relevance. Therefore, in the second optimization step, we train the ETM model in addition to the passage aggregation layer using a *full* document ranking setting, in which there is no passage selection and all passages are scored and the top l passages are chosen for the aggregation layer (i.e., W_{PS}). We initialize the ETM parameters with the best checkpoint obtained from early stopping in the previous optimization step. We again use the binary cross-entropy loss function, this time for a query and a pair of positive and negative documents.

This optimization step further fine-tunes the ETM parameters and learns the W_{PS} parameters.

Step III: Optimizing ESM for Passage Selection using Knowledge Distillation. The last two optimization steps give us an effective document ranking model that runs ETM on every passage in the document and aggregates the scores. In this step, we optimize the ESM parameters. Given the fact that the goal of ESM is to select passages to be consumed by ETM in a cascade setting, we use knowledge distillation for training the ESM model. In other words, we optimize the ESM parameters such that it mimics the ETM behavior using a teacher-student paradigm, where ESM and ETM play the roles of student and teacher, respectively. Therefore, the output of ETM provides labels for the ESM model. A similar idea has been employed in the weak supervision literature [10]. Formally, the loss function for this optimization step is defined as:

$$\mathcal{L}_{\text{Selection}}(q, d) = \mathcal{L}_{\text{KD}}([\text{ETM}(q, p)], [\text{ESM}(q, p)]) \,|\, p \in P \tag{6}$$

where P denotes all the passages in the document d and $\mathcal{L}_{\text{KD}}$ denotes a knowledge distillation loss function. The function $\mathcal{L}_{\text{KD}}$ is responsible for computing the average across passages. A unique feature of our distillation approach is that the teacher signals created by ETM are unsupervised. It is important to train the less capable ESM on the exact distribution it is later used. There are no passage-level labels for all MSMARCO documents we could use to train the ESM, therefore the ETM is the only training signal source.

In our experiments, we study multiple loss functions for knowledge distillation. They include distribution losses, such as mean square error (MSE) and cross entropy, and in-document passage ranking loss functions, such as nDCG2 introduced as part of the LambdaLoss framework [34]. The nDCG2 loss function is a gain-based loss to tightly bind the loss to NDCG-like metrics. For the exact formulation we refer to Wang et al. [34]. In the nDCG2 loss, we assign gain values only to the top k passages sorted by ETM and all other passages receive no gain. The nDCG2 loss, focusing on moving the correct k passages in the top positions is a great fit for our problem: The ESM is only used for pruning or filtering, which means that the ordering inside and outside the top-k set does not matter. The only thing that matters is to find the right set of k passages, as the ETM then creates our final fine-grained scores for each of the k passages. This is a crucial difference in our knowledge distillation approach to concurrent works, which try to fit every decision from the more powerful ranking model to a smaller model [18]. Not surprisingly, we find that using the nDCG2 ranking loss outperforms other loss functions, as discussed in Section 5.1.

3.3 The IDCM Implementation

In this section, we describe the implementation details for the ESM and ETM models used in our experiments.

ESM: The CK Model. The ESM model is the first model in our cascaded architecture and is expected to be extremely efficient. In our experiments, we use CK, an efficient variation of the Conv-KNRM model [9] that combines convolutional neural networks (CNNs) with the kernel-pooling approach of Xiong et al. [36]. Unlike Conv-KNRM that uses multiple convolutional layers with different window sizes for soft-matching of n-grams in the query and document, the CK model uses a single convolutional layer to provide local contextualization to the passage representations without the quadratic time or memory complexity required by Transformer models. In more detail, the CK model transforms the query and passage representations using a CNN layer and uses the cosine function to compute their similarities, which are then activated by Gaussian kernels with different distribution parameters:

$$K_{i,j}^{k} = \exp\left(-\frac{\left(\cos(\text{CNN}(q_i), \text{CNN}(p_j)) - \mu_k\right)^2}{2\sigma^2}\right) \tag{7}$$

where q_i and p_j respectively denote the i^{th} token in the query and the j^{th} token in the passage. μ_k and σ are the Gaussian kernel parameters. Each kernel represents a feature extractor, which is followed by a pooling layer that sums up the individual activations, first by the passage dimension j and then log-activated by the query token dimension i. Each kernel result is weighted and summed with a single linear layer (W_k) as follows:

$$\text{CK}(q, p) = \left(\sum_{i=1}^{|q|} \log\left(\sum_{j=1}^{|p|} K_{i,j}^{k}\right)\right) * W_K \tag{8}$$

ETM: The BERT Ranking Model. Large-scale pre-trained language models, such as BERT [11], have led to state-of-the-art results in a number of tasks, including passage retrieval [23]. The BERT passage ranking model takes sequences representing a query q and a passage p and concatenates them using a separation token. The obtained BERT representation for the first token of the query-passage pair (i.e., the `[CLS]` token) is then fed to a fully-connected layer (W_s) to produce the ranking score:

$$\text{BERT}(q, p) = \underset{\texttt{[CLS]}}{\text{BERT}}(\texttt{[CLS]}; q; \texttt{[SEP]}; p) * W_s \tag{9}$$

where ; denotes the concatenation operation.

4 EXPERIMENT DESIGN

In this section, we describe our experiment setup. We implemented our models using the HuggingFace Transformer library [35] and PyTorch [26]. We employed PyTorch' mixed precision training and inference throughout our experiments for efficiency. In our experiments, we re-rank the documents retrieved by BM25 implemented in Anserini [38]. The query latency measurements are conducted on the same single TITAN RTX GPU.

4.1 Document Collections and Query Sets

For our first passage-training step we utilize the MSMARCO-Passage collection and training data released by Bajaj et al. [1]. We follow the setup of Hofstätter et al. [13] for training the BERT passage ranker. For passage results of the BERT ranking model we refer to the previous work. In all datasets, we limit the query length at 30 tokens to remove only very few outliers, and re-rank 100 documents from BM25 candidates.

We use two query sets for evaluating our models as follows:

TREC DL 2019 and MS MARCO benchmarks. We use the 2019 TREC Deep Learning Track Document Collection [6] that contains 3.2 million documents with a mean document length of 1,600 words and the 80^{th} percentile at 1,900 words. We aim to include them with a 2,000 token limit on our experiments. We selected 5,000 queries from the training set as validation set for early stopping and removed those queries from the training data. We use the following two query sets in our evaluation:

- TREC DL 2019: 43 queries used in the 2019 TREC Deep Learning Track for document ranking task. A proper pooling methodology was used to create a complete set of relevance judgments for these topics [6]. For evaluation metrics that require binary relevance labels (i.e., MAP and MRR), we use a binarization point of 2.
- MS MARCO: 5,193 queries sampled from the Bing query logs contained in the MS MARCO Development set [1]. This query set is larger than the previous one, but suffers from incomplete relevance judgments.

Table 1: Effectiveness results for TREC-DL'19 and MSMARCO DEV query sets. Our aim is to hold the effectiveness of an All-BERT configuration with a cascaded efficiency improvement. * is a stat.sig. difference to All-BERT (2K); paired t-test ($p < 0.05$).

Model	Cascade	# BERT Scored	Doc. Length	TREC DL 2019			MSMARCO DEV		
				nDCG@10	MRR@10	MAP@100	nDCG@10	MRR@10	MAP@100
Baselines									
BM25	–	–	–	0.488	0.661	0.292	0.311	0.252	0.265
TKL [14]	–	–	2K	0.634	0.795	0.332	0.403	0.338	0.345
$\text{PARADE}_{\text{Max-Pool}}$ [18]	–	All	2K	0.666	0.807	0.343	0.445	0.378	0.385
$\text{PARADE}_{\text{TF}}$ [18]	–	All	2K	0.680	0.820	**0.375**	0.446	0.382	0.387
Ours									
IDCM	–	All	512	0.667	0.815	0.348	*0.440	*0.374	*0.383
	–	All	2K	**0.688**	0.867	0.364	**0.450**	**0.384**	**0.390**
	Static First	3	2K	0.638	0.785	0.309	*0.394	*0.330	*0.338
	Static Top-TF	3	2K	*0.624	0.778	0.324	*0.393	*0.329	*0.337
	CK (nDCG2)	3	2K	0.671	0.876	0.361	0.438	0.375	0.380
	CK (nDCG2)	4	2K	**0.688**	**0.916**	0.365	0.446	0.380	0.387

4.2 Training Configuration

We use Adam optimizer [17] with a learning rate of $7 * 10^{-6}$ for all BERT layers. CK layers contain much fewer and randomly initialized parameters and therefore are trained with a higher learning rate of 10^{-5}. We employ early stopping, based on the best nDCG@10 value on the validation set.

4.3 Model Parameters

We use a 6-layer DistilBERT [28] knowledge distilled from BERT-Base on the MSMARCO-Passage collection [13] as initialization for our document training. We chose DistilBERT over BERT-Base, as it has been shown to provide a close lower bound on the results at half the runtime for training and testing [13, 28]. In general our approach is agnostic to the BERT-variant used, when using a language model with more layers and dimensions, the relative improvements of our cascade become stronger, at the cost of higher training time and GPU memory.

For the passage windows we set a base size w of 50 and an overlap of 7 for a total window size of 64. In a pilot study, we confirmed that a larger window size does not improve the All-BERT effectiveness results. For the BERT passage aggregation, we use the top 3 BERT scores to form the final document score, independent of the cascade selection count. This allows us to base all selection models on the same All-BERT instance.

We set the token context size for CK to 3 and evaluate two different CK dimensions, first a full 768 channel convolution corresponding to the dimensions of the BERT embeddings and second a smaller convolution with a projection to 384 dimensions before the convolution and another reduction in the convolution output dimension to 128 per term, which we refer to as CKS.

5 RESULTS

In this section, we address the research questions raised in Section 1.

5.1 RQ1: Knowledge Distillation and Effectiveness Study

Our first research question centers around training techniques and we investigate:

RQ1 Can IDCM achieve comparable effectiveness to the full BERT-based ranker at lower computation cost and query latency?

To address this research question, we compare the proposed method to a number of strong retrieval baselines, including BM25, TKL [14], and PARADE [18]. TKL is a non-BERT local self-attention ranking model with kernel-pooling; ANCE Max-P is a dense, single vector for query and passages, ranker trained with an iterative negative sampling strategy; $\text{PARADE}_{\text{MAX}}$ is very close to our All-BERT baseline, in that it scores every passage with a BERT ranker and aggregates passage representations in a lightweight layer. The results are reported in Table 1. The first observation on the All-BERT setting of IDCM, without cascading, is the strong difference in effectiveness across collections between 512 document tokens and 2K tokens. The All-BERT 2K model outperforms all previously published single model re-ranking baselines on the TREC DL 2019 dataset, even with DistilBERT as the language model. We see how training and evaluating with longer document input improves the effectiveness. This is also a strong argument for our cascading approach that makes it possible to process long inputs. We chose the All-BERT 2K setting as the base model for all following results.

Furthermore we use static cascade selectors as baselines, that use BERT scoring after static selections. One selects the first 3 passages by position, to have a direct comparison to CK selections. Another option we benchmark is the use of raw term frequency matches, where we take the three passages with the highest frequency of direct term matches without semantic or trained matching. Both approaches fail to improve the scoring over the baselines or our CK selection model. The term frequency selection even underperforms the first positions selector.

Our main IDCM configuration with a knowledge distilled CK shows strong results, which are not statistically different to their

Table 2: Impact of knowledge distillation with measures using a cutoff at 10. * indicates stat.sig. difference.

Scoring Model	Training	TREC-DL'19		MSMARCO	
		nDCG	MRR	nDCG	MRR
CK	Standalone	0.551	0.677	0.353	0.287
CK	BERT-KD	*0.595	0.749	*0.363	*0.299

Table 3: Knowledge distillation loss study of the IDCM cascade training with measures using a cutoff at 10. Stat.sig. is indicated with the superscript to the underlined character.

# BERT Scored	KD-Loss	TREC-DL'19		MSMARCO	
		nDCG	MRR	nDCG	MRR
3	MSE	0.664	0.816	0.426	0.362
	Cross Entropy	0.667	0.851	0.437	0.373
	nDCG2	**0.671**	**0.876**	**0.438**	**0.375**
4	MSE	0.683	0.870	0.437	0.374
	Cross Entropy	0.675	0.889	^{m}**0.446**	^{m}**0.381**
	nDCG2	**0.688**	**0.916**	^{m}**0.446**	^{m}0.380

base model of All-BERT (2K). Across both query sets, the select 3 is already close to the reachable results and on select 4 we obtain better MRR and MAP results on TREC-DL'19, even though they are not significantly different.

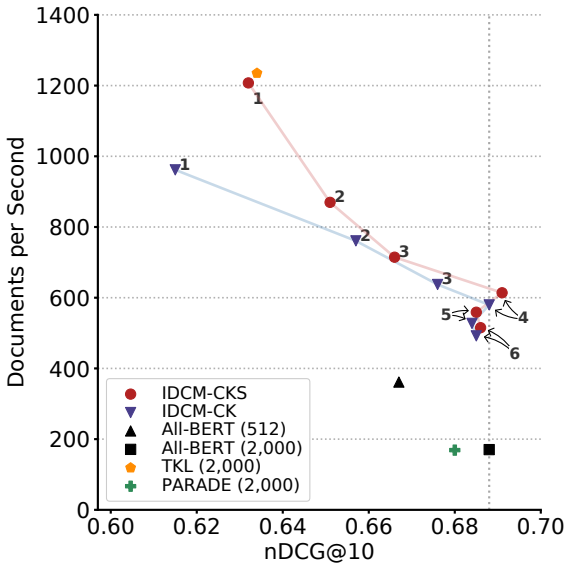
We further compare two training strategies for the efficient CK model: (1) knowledge distillation as described in Section 3.2, and (2) standalone training on relevance judgments. The results are reported in Table 2. We observe that the knowledge distilled CK model leads to significantly higher performance on both TREC DL 2019 and MS MARCO datasets in comparison to the standalone training. The improvements on the TREC DL 2019 query set are much larger. As expected, CK alone shows substantially lower effectiveness compared to BERT. Overall we show that even though alone CK is ineffective, combined with BERT in the IDCM model produces very effective and efficient results.

We extend our analysis by inspecting different knowledge distillation losses and their effect on the full cascaded setting in Table 3. We probe knowledge distillation with an MSE, Cross Entropy and the LambdaLoss nDCG2 losses for two different cascade selection numbers, three and four. When four passages are selected, all three knowledge distillation losses perform on par with the All-BERT model. Overall using nDCG2 as the knowledge distillation loss outperforms the other two loss functions and is the most stable across the two evaluated query sets. Therefore, in the following experiments, we only focus on nDCG2 as the default knowledge distillation loss.

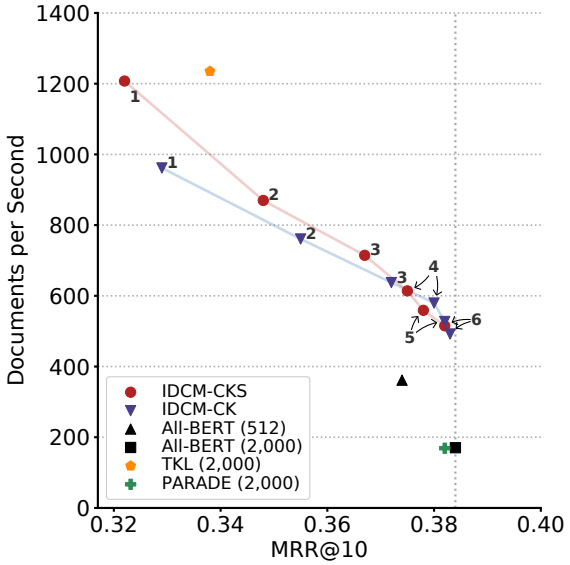
5.2 RQ2: Efficiency-Effectiveness Analysis

To study our next research question we turn to the viewpoint of both efficiency and effectiveness of different IDCM model configurations to answer:

RQ2 How is the effectiveness-efficiency trade-off influenced by the number of passages k that the less expensive model selects from the document?



(a) Throughput and nDCG@10 results on TREC DL 2019.



(b) Throughput and MRR@10 results on MS MARCO Dev.

Figure 3: Throughput and ranking effectiveness trade-off results. The vertical line shows the achievable effectiveness, for all IDCM models based on All-BERT (2,000). The number next to the marker indicates the selection count.

The selection variants of IDCM are all based on the All-BERT setting and therefore this model instance sets our potential for cascaded effectiveness results, as we do not train the BERT module during selection training. In Figures 3a and 3b we compare the model throughput of documents per second (y-axes) with their effectiveness (x-axes). Akin to a precision-recall plot, the best result would be situated in the upper right corner.

We evaluated IDCM's selection parameter – the number of passages scored with the costly BERT model – k from 1 to 6 using the

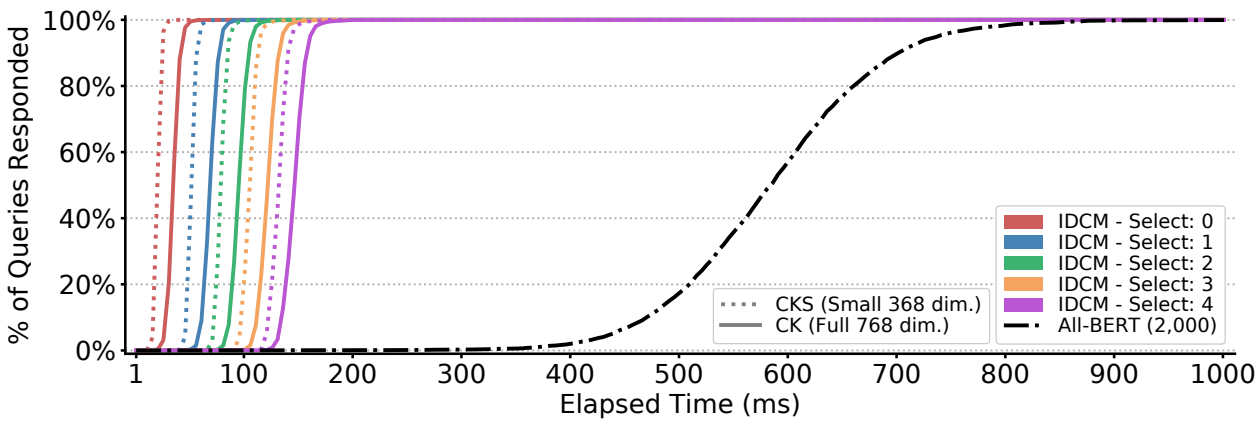


Figure 4: Fraction of queries that can be answered in the given time-frame for re-ranking 100 documents with up to 2,000 tokens on MSMARCO. Select 0 means only CK timing without BERT cascading.

full convolution CK and a smaller dimensional CKS setting. We find that selecting too few passages reduces the effectiveness strongly, however starting with a selection of 4 or more passages, IDCM results are very similar to All-BERT results, while providing a much higher throughput. On the nDCG@10 metric of TREC-DL'19 in Figure 3a IDCM reaches the All-BERT effectiveness starting with 4 selected passages. In Figure 3b the IDCM setting is already close to the reachable effectiveness, and taking 5 and 6 passages close the gap to All-BERT further, to a point of almost no differences.

A simple efficiency improvement is to use a lower document length on All-BERT passage scoring, such as limiting the document to the first 512 tokens. We find that this works both more slowly and less effectively than IDCM.

It is important to note, that in all our experiments presented here we utilize the 6-layer DistilBERT encoder. When we compare with related work, which commonly uses larger BERT-style models, such as the original BERT-Base or BERT-Large, we show even better efficiency improvements. A 12-layer BERT-Base model in the All-BERT (2,000) configuration can only process 85 documents per second, and the 24-layer BERT-large only manages to score 30 documents per second. Applying our IDCM technique to these larger encoders, brings even larger performance improvements.

In summary, with this analysis we show how IDCM can be as effective as an All-BERT model, while maintaining a four times higher throughput. In addition, users of the IDCM model have the option to trade-off effectiveness for more efficiency, along a clear curve.

5.3 RQ3: Query Latency Analysis

The mean or median aggregations of query latencies hide crucial information about the tail of queries that require more processing time. In the case of neural ranking models this tail is heavily dependent on the total document length of all documents in a batch. We now study efficiency in detail with:

RQ3 How does IDCM compare to the baseline with respect to variance in query latency?

In Figure 4 we plot the fraction of queries (y-axis) that can be processed by the neural models in the given time (x-axis). The

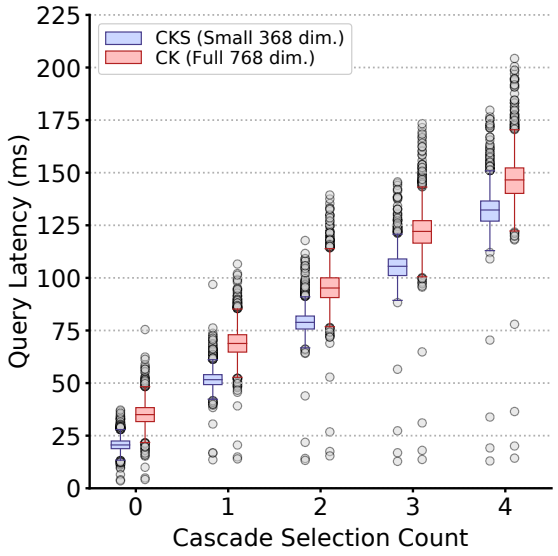


Figure 5: Query latency for different IDCM cascade selection configurations for re-ranking 100 documents with up to 2,000 tokens on MSMARCO. Selection 0 only measures the time for the CK module without routing passages to BERT.

All-BERT baseline (dash-dotted black line) for documents up to 2,000 tokens has a large range in the time required to re-rank documents. The All-BERT setting of IDCM already is a strong baseline for query latency, as we compact passage representations, to skip padding-only passages. However, it still requires BERT to run on up to 40 passages. Our IDCM model configurations with two CK sizes (dotted and full line) show a much lower variance and overall faster query response. Naturally, the more passages we select for cascading to the BERT module, the longer IDCM takes to compute.

Now, we turn to a more focused query latency view of the different IDCM configurations with boxplots of the query latency distributions in Figure 5. Again, we report timings for a full CK (red; right side) and smaller CKS variant (blue; left side). The first entry with a selection of 0 shows only the computational cost of

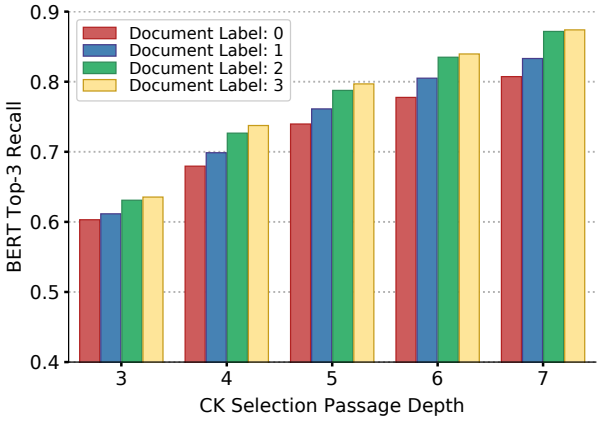


Figure 6: Intra-document passage selection recall of the CK selection module in comparison to the top-3 BERT selection split by document relevance grade on TREC-DL'19. The higher the label the more relevant a document.

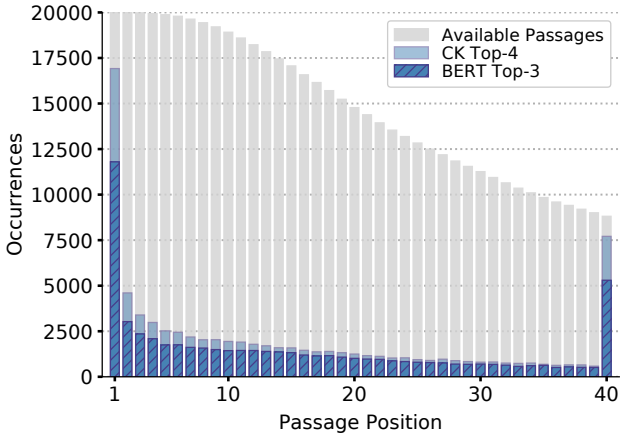


Figure 7: Selected Passages by the CK top-4 module and then subsequently scored by BERT for the top-3 scoring.

the CK module for 2,000 tokens without running BERT. When we compare the latency differences between 0 and 1 selection we can see that computing BERT for a single passage per document adds 25 ms to the median latency. This shows the efficiency of CK(S): roughly 40 passages processed with CK have an equal runtime compared to a single passage that BERT requires.

5.4 RQ4: Passage Selection Analysis

As presented in Section 3.2 we aim to train the CK module to imitate the passage scoring of the BERT module. To understand how well the CK model is able to do just that we evaluate the intra-document passage recall of the CK scores in comparison to the top-3 BERT passages selected to form the final document score and study:

RQ4 How often does the passage selection under IDCM recall the same passages as those scored highly by the BERT model?

In Figure 6 we plot the CK recall for different numbers of selected CK-passages and split the reporting by document relevance grades on the TREC-DL'19 query set. We find an interesting pattern among the different relevance classes: CK is able to provide more accurate passage selections the more relevant a document is.

A recall of 1 would guarantee the same document score, however as we showed it is not necessary for the IDCM model to provide very close effectiveness results to the original ranking.

Finally, we inspect the behavior of both BERT and CK modules with respect to the positions of the highest scoring passages. In Figure 7 we investigate the top selections of both CK and BERT along the positions of the passages. In gray in the background are the available passages per position of the top-100 documents of the TREC-DL'19 query set, they reduce as not all documents are long enough to fit the maximum used length of 2,000 tokens. The All-BERT setting needs to compute BERT scores for all available passages, whereas in IDCM the selected passages in blue are the only passages score by BERT. The top-3 passages from BERT are furthermore visualized by the shaded dark blue bars.

We see a strong focus on the first and last possible passages selected by the modules. A focus on the first passages is to be expected as the title and introduction of web pages is situated there, however the focus on the last passages is more curious. Because of our symmetric padding, the first and last passages have an empty (padded) overlap. This is the only indicator to the BERT model of passage positions in a document, the absence of 7 more tokens. It seems this is the signal the BERT model picks up on and subsequently trains the CK model to follow along. We leave deeper investigations of the behavior of BERT in different document parts for future work and conclude this section with the finding that IDCM learns to use all available passages including the end of a document input.

6 CONCLUSION

Applying a BERT model many times per document, once for each passage, yields state-of-the-art ranking performance. The trade-off is that inference cost is high due to the size of the model, potentially affecting both the mean and variance of query processing latency. Typically in neural retrieval systems, one is forced to a make a clear decision between reaching efficiency or effectiveness goals. In this work we presented IDCM, an intra-document cascaded ranking model that provides state-of-the-art effectiveness, while at the same time improving the median query latency by more than four times compared a non-cascaded full BERT ranking model. Our two-module combination allows us to efficiently filter passages and only provide the most promising candidates to a slow but effective BERT ranker. We show how a key step in achieving the same effectiveness as a full BERT model is a knowledge distilled training using the BERT passage scores to train the more efficient selection module. Our knowledge distillation provides self-supervised teacher signals for all passages, without the need for manual annotation. Our novel distillation technique not only improves the query latency of our model in a deployment scenario, it also provides efficiency for replacing manual annotation labor and cost with a step-wise trained teacher model. In the future we plan to extend the concept of intra-document cascading for document ranking to a dynamic number of passages selected and more cascading stages.

REFERENCES

[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew Mcnamara, Bhaskar Mitra, and Tri Nguyen. 2016. MS MARCO : A Human Generated MAchine Reading COmprehension Dataset. In *Proc. of NIPS*.

[2] Michael Bendersky and Oren Kurland. 2008. Utilizing passage-based language models for document retrieval. In *Proc. of ECIR*.

[3] Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *MSR-Tech Report* (2010).

[4] James P Callan. 1994. Passage-level evidence in document retrieval. In *Proc. of SIGIR*.

[5] Ruey-Cheng Chen, Luke Gallagher, Roi Blanco, and J Shane Culpepper. 2017. Efficient cost-aware cascade ranking in multi-stage retrieval. In *Proc. of SIGIR*.

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

[7] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2019. Overview of the trec 2019 deep learning track. In *TREC*.

[8] Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for IR with contextual neural language modeling. In *Proc. of SIGIR*.

[9] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search. In *Proc. of WSDM*.

[10] Mostafa Dehghani, Arash Mehrjou, Stephan Gouws, Jaap Kamps, and Bernhard Schölkopf. 2018. Fidelity-weighted learning. *Proc. of ICLR* (2018).

[11] J. Devlin, M. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of NAACL*.

[12] Luke Gallagher, Ruey-Cheng Chen, Roi Blanco, and J Shane Culpepper. 2019. Joint optimization of cascade ranking models. In *Proc. of WSDM*.

[13] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. arXiv:cs.IR/2010.02666

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

[15] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351* (2019).

[16] Marcin Kaszkiel and Justin Zobel. 1997. Passage retrieval revisited. In *ACM SIGIR Forum*, Vol. 31. ACM New York, NY, USA, 178–185.

[17] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[18] Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2020. PARADE: Passage Representation Aggregation for Document Reranking. *arXiv preprint arXiv:2008.09093* (2020).

[19] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2020. Pretrained Transformers for Text Ranking: BERT and Beyond. *arXiv preprint arXiv:2010.06467* (2020).

[20] Xiaoyong Liu and W Bruce Croft. 2002. Passage retrieval based on language models. In *Proc. of CIKM*.

[21] Irina Matveeva, Chris Burges, Timo Burkard, Andy Laucius, and Leon Wong. 2006. High accuracy retrieval with multiple nested ranker. In *Proc. of SIGIR*.

[22] Bhaskar Mitra, Sebastian Hofstatter, Hamed Zamani, and Nick Craswell. 2020. Conformer-Kernel with Query Term Independence for Document Retrieval. *arXiv preprint arXiv:2007.10434* (2020).

[23] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).

[24] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with BERT. *arXiv preprint arXiv:1910.14424* (2019).

[25] R. Nogueira, W. Yang, J. Lin, and K. Cho. 2019. Document Expansion by Query Prediction. *arXiv preprint arXiv:1904.08375* (2019).

[26] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *Proc. of NIPS-W*.

[27] Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

[28] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[29] Eilon Sheetrit, Anna Shtok, and Oren Kurland. 2020. A passage-based approach to learning to rank documents. *Information Retrieval Journal* (2020), 1–28.

[30] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136* (2019).

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, et al. 2017. Attention is all you need. In *Proc. of NIPS*.

[32] Paul Viola and Michael Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *Proc. of CVPR*.

[33] Lidan Wang, Jimmy Lin, and Donald Metzler. 2011. A cascade ranking model for efficient ranked retrieval. In *Proc. of SIGIR*.

[34] Xuanhui Wang, Cheng Li, Nadav Golbandi, Mike Bendersky, and Marc Najork. 2018. The LambdaLoss Framework for Ranking Metric Optimization. In *Proc. of CIKM*.

[35] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv* (2019), arXiv–1910.

[36] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *Proc. of SIGIR*.

[37] Ming Yan, Chenliang Li, Chen Wu, Bin Bi, Wei Wang, Jiangnan Xia, and Luo Si. 2019. IDST at TREC 2019 Deep Learning Track: Deep Cascade Ranking with Generation-based Document Expansion and Pre-trained Language Modeling.. In *TREC*.

[38] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of Lucene for information retrieval research. In *Proc. of SIGIR*.

[39] Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Cross-domain modeling of sentence-level evidence for document retrieval. In *Proc. of EMNLP-IJCNLP*.

[40] Hamed Zamani and Nick Craswell. 2020. Macaw: An Extensible Conversational Information Seeking Platform. In *Proc. of SIGIR*.