RESEARCH-ARTICLE

Efficient Re-ranking with Cross-encoders via Early Exit

FRANCESCO BUSOLIN, Ca' Foscari University of Venice, Venice, VE, Italy

CLAUDIO LUCCHESE, Ca' Foscari University of Venice, Venice, VE, Italy

FRANCO MARIA NARDINI, Institute of Information Science and Technologies "Alessandro Faedo", Pisa, PI, Italy

SALVATORE ORLANDO, Ca' Foscari University of Venice, Venice, VE, Italy

RAFFAELE PEREGO, Institute of Information Science and Technologies "Alessandro Faedo", Pisa, PI, Italy

SALVATORE TRANI, Institute of Information Science and Technologies "Alessandro Faedo", Pisa, PI, Italy

View all

Published: 13 July 2025

SIGIR '25: The 48th International ACM SIGIR Conference on Research and Development in Information Retrieval
July 13 - 18, 2025
Padua, Italy

Conference Sponsors:
SIGIR

https://doi.org/10.1145/3726302.3729962
ISBN: 9798400715921

Efficient Re-ranking with Cross-encoders via Early Exit

Francesco Busolin
francesco.busolin@unive.it
Ca' Foscari University of Venice
Venice, Italy

Claudio Lucchese
claudio.lucchese@unive.it
Ca' Foscari University of Venice
Venice, Italy

Franco Maria Nardini
francomaria.nardini@isti.cnr.it
ISTI-CNR
Pisa, Italy

Salvatore Orlando
orlando@unive.it
Ca' Foscari University of Venice
Venice, Italy

Raffaele Perego
raffaele.perego@isti.cnr.it
ISTI-CNR
Pisa, Italy

Salvatore Trani
salvatore.trani@isti.cnr.it
ISTI-CNR
Pisa, Italy

Alberto Veneri
alberto.veneri@unive.it
Ca' Foscari University of Venice
Venice, Italy

Abstract

Pre-trained language models based on transformer networks are highly effective for document re-ranking in ad-hoc search. Among these, *cross-encoders* stand out for their effectiveness, as they process query-document pairs through the entire transformer network to compute ranking scores. However, this traversal is computationally expensive. To address this, prior work has explored early-exit strategies, enabling the model to terminate the traversal of query-document pairs. These techniques rely on learned classifiers, placed after each transformer block, that decide if a query-document pair can be dropped. Diverging from previous approaches, we propose Similarity-based Early Exit (SEE), a novel—non-learned—strategy that exploits the similarities between query and document token embeddings to early-terminate the inference of documents that will most likely be non-relevant to the query. Even though SEE can be used after every transformer block, we show that the best advantage is achieved when applied before the first transformer block, thus saving most of the inference cost for the query-document pairs. Reproducible experiments on 17 public datasets covering in-domain and out-of-domain evaluation show that SEE can be effectively applied to four different cross-encoders, achieving speedups of up to 3.5× with a limited loss in ranking effectiveness.

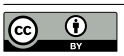
CCS Concepts

• **Information systems** → **Language models**; **Retrieval efficiency**.

Keywords

Early Exit, LLM-based rankers, Efficiency

ACM Reference Format:
Francesco Busolin, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, Salvatore Trani, and Alberto Veneri. 2025. Efficient Re-ranking with Cross-encoders via Early Exit. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3726302.3729962



SIGIR '25, Padua, Italy

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3729962

1 Introduction

Pre-trained Language Models (PLMs) based on transformer networks achieve state-of-the-art performance in ad-hoc search. The drawback of using these complex architectures is their high computational cost, so balancing efficiency with ranking quality is an ongoing challenge [25, 4]. A common approach to reduce overall computational requirements is to use PLMs for *re-ranking* only. This involves applying them after a first-stage retrieval—e.g., using term-based models like BM25—to narrow down the set of potentially relevant documents to which the PLM is applied [17, 13]. Some PLMs, commonly referred to as *cross-encoders* in the literature, process query-document pairs as input. Their embeddings pass through the transformer's all-to-all interaction blocks to produce a re-ranking score, which is then used to generate the final results list.

While the research community has extensively explored model compression techniques like distillation and quantization to reduce cross-encoder computational costs, early-exit strategies in neural document re-ranking have received relatively little attention [47, 20]. Most of the contributions in literature focusing on early-exiting transformer-based networks revolves around supervised techniques, i.e., each transformer layer of the PLM is coupled with a classifier that implements the early exit decision.

For instance, let us consider DeeBERT by Xin *et al.*[48], which aims to accelerate BERT[12] inference. In DeeBERT, when a sample goes through a transformer block, it is also passed to the associated classifier. This classifier is fine-tuned alongside the entire network during training. This approach is illustrated in the *top* diagram of Fig. 1, where the arrows represent the flow of the embeddings encoding a given query-document pair. These embeddings pass through the transformer blocks T_i of the PLM to compute a final relevance score. Notably, they can exit early at any block if the intermediate classifier is sufficiently confident in its predicted score.

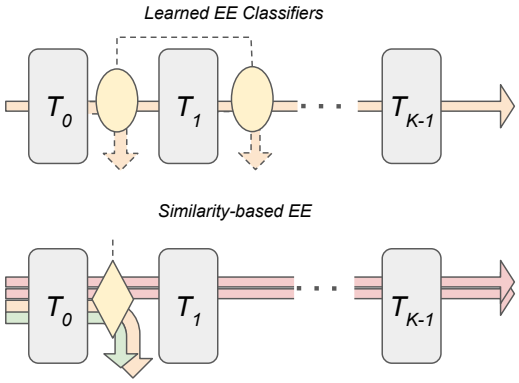


Figure 1: *Top*: Architecture of a learned, point-wise Early Exit (EE) method based on multiple classifiers. *Bottom*: Architecture of our SEE method that early-stops non-relevant documents with a single listwise filter.

In this paper, we propose a new class of *Early Exit (EE)* strategies for cross-encoders. Our approach builds on recent studies that analyze the properties of this family of models [27, 10], particularly the evolution of token embeddings in their latent space as they pass through the cross-encoder blocks. Based on these insights, we introduce a novel, non-learned early-exit technique, named Similarity-based Early Exit (SEE), which leverages *similarities between embeddings* as a proxy for the final relevance score computed by the PLM. This proxy enables the discrimination of documents likely to be relevant to the query from those that are not. Relevant documents continue traversing the transformer blocks to refine their final score, while non-relevant ones exit the inference process early, maintaining the quality of the final top-k result list.

Unlike learned approaches, where the early-exit decision is *point-wise*—i.e., made independently for each query-document pair—SEE takes exit decisions in a *listwise* manner. It employs a *filter*, placed before a specified transformer block, which considers all documents to be re-ranked for a given query. The *bottom* diagram in Fig. 1 illustrates how SEE operates, using an example where the filter is positioned before the transformer block T_1. For a given query, the filter processes all documents to be re-ranked, reordering them based on embedding similarities. It then outputs only the top-ranked documents, allowing them to continue traversal through the transformer network. When the filter is placed before the first transformer block (T_0), SEE consistently achieves the best efficiency-effectiveness trade-off. Specifically, our evaluation shows that SEE outperforms state-of-the-art methods, achieving speedups of up to 3.5× compared to the full model, with only a limited loss in ranking effectiveness.

In detail, the **novel contributions** presented are the following:

- We propose SEE, an effective, **non-learned EE strategy** for PLM-based cross-encoders that leverage properties of the embedding space during inference. To the best of our knowledge, this is the first investigation of early-exit strategies for PLM-based re-rankers that do not require end-to-end training of additional classifiers explicitly designed for early exit query-document inference.
- We present a detailed comparison of SEE applied to MonoBERT against its state-of-the-art competitor, early-exiting MonoBERT (eeMB) [47], in an **in-domain** scenario on 3 different datasets. The analysis shows that SEE can achieve comparable performance with respect to eeMB without additional training. To allow the best possible fairness of the experimental protocol, we provide an improved GPU implementation of eeMB, which substantially improves its inference efficiency.
- We provide an exhaustive evaluation of SEE in an **out-of-domain** setting over 14 different datasets, showing that it outperforms eeMB on most of them, achieving speedups of up to 3.5× with minimal loss in ranking effectiveness.
- We experimentally assess that SEE can also be used in combination with other inference-optimization techniques such as Ranked List Truncation (RLT). In particular, we show that SEE is **robust to different document list sizes** to be re-ranked, with sizes ranging from 100 to 1000, providing a speedup of 1.4× even on short-sized lists.
- We analyze the **adaptability** of SEE by implementing it in 3 additional state-of-the-art cross-encoder, i.e., MonoELECTRA, miniLM, and X-ELECTRA, showing that the performance of our proposed method is consistent across different models.

2 Related Work

Researchers have delved into the efficiency/effectiveness tradeoff of Information Retrieval (IR) query processing by exploring various directions [5, 37, 15]. The recent developments of Language Model (LM) based on pre-trained transformers, used in IR for retrieval and ranking, have posed interesting challenges and opportunities in this direction.

Although the advantages in the effectiveness of PLM-based cross-encoders are well known, their high computational requirements of the inference step can impact the efficiency of the overall query processing pipeline. For this reason, in recent years, several research lines have contributed to improving the efficiency of the inference phase of PLM models. Most apply *model compression* [50] techniques. More specific examples of these techniques are: *model distillation* [19], *model quantization* [31], and *early exit* (EE) [48]. Unlike the first two lines of research, which work by speeding up the computation time of PLM models by reducing their size, the third line actively reduces the computational burden of PLMs by introducing internal checkpoints (also called early exit *filters*) that actively drop the inference of a specific sample when its likelihood of being relevant is low. In addition, EE can be combined with distillation and quantization to further speed up inference. However, the idea of EE for machine learning inference was originally developed many years before modern PLMs and then applied to IR. Notably, in IR these EE techniques were applied to learning-to-rank models, such as additive tree ensembles [8, 6, 7].

Several contributions in literature investigate EE techniques for PLM-based cross-encoders, revolving around the same concept, i.e., each transformer layer of the PLM is coupled with a classifier that implements the EE filter. One of the first implementations of this idea is due to Xin *et al.* [48], who proposed DeeBERT to speed up the inference of BERT [12]. Transformer layers and associated classifiers are jointly fine-tuned on a given downstream task. At

inference time, after a sample goes through a transformer layer, it is also passed to the associated classifier. If it is confident of the prediction, the result is returned; otherwise, the sample is sent to the next transformer layer.

Zhou *et al.* proposed a more conservative EE approach by introducing *Patience-based Early Exit* (PABEE) for BERT [52]. Still, a classifier is inserted after each layer of a BERT-based PLM, and inference stops when the intermediate predictions of the internal classifiers remain unchanged for a predefined number of steps.

Xin *et al.* [47] proposed eeMB (early-exiting MonoBERT), which is the first EE technique applied to PLM-based cross-encoders for document ranking. Although the general idea developed in eeMB is the same as that of DeeBERT, the authors apply it to MonoBERT. Due to the task addressed, the classifiers used to early-exit the computation by performing a binary classification on query–document pairs into relevant/non-relevant, and the inference stops when a classifier after a transformer block is confident in its prediction. Moreover, unlike DeeBERT, which treats all classes equally, the authors use asymmetric EE for document ranking: the exiting *confidence threshold* for positive predictions (relevant documents) is higher than for negative ones (non-relevant document), since the two classes in document ranking are intrinsically different. The main hyper-parameter of eeMB is thus c_n, the *negative confidence threshold*, used to EE samples that are likely non-relevant.

In this paper, we evaluate the performance of SEE against eeMB by comparing their effectiveness/efficiency trade-offs. Besides eeMB, more sophisticated learned EE strategies have been proposed, such as BERxiT [49], which employ a learned approach similar to the one proposed by eeMB, i.e., it exploits an EE strategy in which a supervised classifier acts as a filter. Unlike eeMB, BERxiT does not address the ranking task. To the best of our knowledge, eeMB is the only EE method that can be exploited in an ad-hoc re-ranking task with point-wise cross-encoders. Note that, as in previous work in the literature, this work focuses on—point-wise—encoder-only models, i.e., MonoBERT, MonoELECTRA, miniLM, and X-ELECTRA. We do not consider more recent—listwise—PLM-based cross-encoders, such as RankZephyr [32], LiT5 [39] and Set-Encoder [38] because experimental results in literature show that they are as effective as pointwise re-rankers [38].

In addition to EE strategies, RLT, a.k.a. query cut-off prediction [29], techniques are related to our proposed method. While in a RLT technique, a speedup is achieved by reducing the number of documents to be re-ranked as in a EE strategy, the decision on re-ranking or not a given document depends on the score of the retrieval system used and not on the re-ranking models employed. This fundamental difference allows us to use a RLT technique and a EE strategy in the same ranking pipeline. In this paper, we also present an experimental evaluation that shows the impact of applying RLT before the EE strategy.

Unlike previous approaches in the literature, our SEE technique leverages the intrinsic properties of the embeddings to guide EE strategies. Specifically, instead of modifying the cross-encoder by introducing additional learned parameters for EE, we exploit the inherent characteristics of the embedding space and its spatial distribution. This enables us to devise a similarity-based technique that operates without any modifications to the original cross-encoder model.

3 Similarity-based Early Exit

Given the recent results highlighting the importance of analyzing the representation learned by PLMs for automatic text classification [10] and document ranking [27], we hypothesize that token representations can be exploited also to early exit samples in a document re-ranking scenario. Specifically, we investigate whether a proper similarity measure between query and document token embeddings can be used as a proxy of the final document score: the smaller the similarity, the lower the probability that the document will be scored high by the cross-encoder. Under this hypothesis, we can evaluate query-document similarity after any transformer block and filter in advance documents with low similarity to save the cost of traversing the remaining transformer blocks.

3.1 Similarity Measures

We investigate four similarity measures as proxies of the ranking score computed by a cross-encoder model for a given query-document pair (q, d). Let $\{q_i\}_{i=1}^{N}$ and $\{d_j\}_{j=1}^{M}$ be the embeddings for tokens of a query q and a document d, respectively. In this work, we always refer to embeddings before a specific transformer block T_i, $0 \leq i < K$, with K the number of transformer blocks in the neural network. According to this notation, q_i and d_j before T_0 correspond to the input embeddings of the model. In the base version of BERT-based models, we have 12 transformer blocks, and thus $K = 12$.

Given an embedding similarity measure $\phi(q_i, d_j)$, e.g., cosine similarity, we consider the following ways of aggregating the similarities between the two sets $\{q_i\}$ and $\{d_j\}$:

$$\textsc{Max}(q, d) = \max_{\substack{1 \leq i \leq N \\ 1 \leq j \leq M}} \phi(q_i, d_j) \tag{1}$$

$$\textsc{MeanSim}(q, d) = \frac{1}{N \cdot M} \sum_{i=1}^{N} \sum_{j=1}^{M} \phi(q_i, d_j) \tag{2}$$

$$\textsc{CentrSim}(q, d) = \phi\left(\frac{1}{N} \sum_{i=1}^{N} q_i, \frac{1}{M} \sum_{j=1}^{M} d_j \right) \tag{3}$$

$$\textsc{MaxSim}(q, d) = \sum_{i=1}^{N} \max_{1 \leq j \leq M} \phi(q_i, d_j) \tag{4}$$

Max (Eq. 1) measures the maximum similarity between a query's and a document's embedding. It is sufficient to have a single pair of embeddings (q_i, d_j) with high similarity to conclude that q and d are similar. MeanSim (Eq. 2) is a more conservative estimate of similarity, where the average distance between all possible pairs of query/document embeddings is considered. CentrSim (Eq. 3) provides a behavior similar to MeanSim at a reduced cost: only the distance between the centroids of the two sets is computed. Note that the above three measures are borrowed from the hierarchical agglomerative clustering linkage measures [34]. Finally, it is straightforward to see that Max is easily influenced by outlier token embeddings, leading to a similarity over-estimation. In contrast, high values of the other two measures are obtained only if the two sets $\{q_i\}$ and $\{d_j\}$ are globally similar. To overcome these issues, we also consider the MaxSim (Eq. 4) similarity measure, which was

introduced by Khattab and Zaharia [22] for ColBERT and its variants [36, 35]. Intuitively, for each query token embedding, MaxSim searches for the most similar document token embedding and then aggregates similarities through summation. Thus, one similar pair is no longer sufficient to consider q and d similar.

Recall that token embeddings change during the traversal of the transformer blocks of the network. In principle, our early-exit filter can compute the above similarities before any transformer block. The earlier we can discard a document, the more significant the gain in efficiency. However, intuition suggests that the more blocks are evaluated, the more *refined* are the embeddings, and therefore, the more reliable the conclusions we can draw by considering their similarity. However, we have to consider that embeddings are transformed during the network traversal, and the boundary between query and document tokens, on which our similarity measures are based, might become blurry due to the token interaction caused by the attention layers. Despite this issue, we still adopt a simplified—yet conservative—approach in computing our similarity measures. We maintain the same distinction between query and document token embeddings, adopted to feed the model and also for all the transformed embeddings obtained during the network traversal. This follows common intuition and simplifies our goal of understanding whether the similarity measures in Eq. 1-4 are good proxies for the ranking scores predicted by PLM-based cross-encoders models. We experimentally assess this assumption in Sec. 4.

3.2 SEE Architecture and Strategies

In this section, we discuss SEE, a novel method that aims to early exit the inference process of cross-encoder models. SEE directly exploits the similarity between token embeddings to decide whether to exit the inference of query-document pairs. As already mentioned, we focus on the re-ranking task, where we have a user query q and a list of documents $L_q = [d_1, d_2, \ldots, d_{|L_q|}]$ to be re-ranked, as identified by a first-stage retriever.

Early Exit Architecture. SEE works as a filter that can be applied before any transformer block of a cross-encoder. The SEE filter first estimates query-document similarities for all pairs $\{(q, d_i)\}_{i=1,\ldots,L_q}$, employing one of the similarity functions previously introduced. Then, it reorders the list L_q, using these query-document similarities as proxies of the final cross-encoder scores. Finally, it truncates the reordered list L based on a similarity threshold τ. If the similarity measure for a given query-document pair (q, d_i) is below τ, we stop scoring (q, d_i), thus making τ the trigger of our early-exit decision. Otherwise, the pair will continue the traversal of the model up to the end, as done when using the original cross-encoder model without an early exit. Note that in the way we defined our EE approach, the number of query-document pair (q, d_i) completing the entire traversal of the model is not fixed since in different queries, a difference ratio of query-document pairs will pass the similarity filter.

The efficiency/effectiveness trade-off of SEE relies on a proper choice of τ, which does not require a training phase. Once identified a query-document similarity function, the choice of τ is made using some heuristic strategies, adapted from those introduced by Cambazoglu *et al.* [8] for early exit document ranking in the context of Learning-to-Rank methods based on additive tree ensembles. Therefore, no learning is needed to tune τ, which is a major difference from current state-of-the-art approaches for early exit for cross-encoder inference that exploits multiple learned classifiers.

In addition, unlike previous learned approaches [47, 49] that are point-wise, because early-exit decisions are taken independently of all other query-document pairs, SEE can be considered listwise, as the list of documents L_q is first reordered in reverse order by similarity values to decide the threshold τ that triggers our early-exit strategy.

Finally, it is worth noting that SEE, is particularly well suited for performing inference on large batches of query-document pairs, as is common practice for fully exploiting the power of high-end GPUs. Indeed, we can split the original model into two separate "slices": *(i)* the first from the input embedding to the filter, *(ii)* the second from the filter until the final output. Only one *re-batch* step is required in between the two phases to handle the shortening of the original list L_q, which largely reduces the number of query-document pairs that have to continue the cross-encoder inference to obtain their final scores. The same cannot be easily done in previous early-exit approaches, where a point-wise classifier is implemented after each transformer block, and thus a re-batching should occur after each block to handle the progressive shortening of L_q.

Early Exit Strategies. SEE is based on query-document similarities, which are normalized in the range $[0-1]$, aiming to compare different similarity values based on a single scale. The min-max normalization of similarities is carried out on a per query basis, and thus we independently scale in $[0-1]$ the similarities computed for all documents in each L_q.

Several strategies can govern the choice of τ for early-exiting query-document pairs. The first strategy is based on a fixed similarity threshold τ, which is made possible due to the scaling of the similarity values in $[0-1]$. Independent of the query and the document, the same value of τ is used to decide whether to continue the scoring. For example, we can set $\tau = 0.8$, and a document is filtered out only if its normalized similarity is less than 0.8. We refer to this strategy as Exit by Similarity Threshold (EST). The higher the value of τ, the stricter our SEE filter becomes.

A second strategy, named Exit by Proximity Threshold (EPT), considers the objective of enhancing and optimizing ranking at a fixed cutoff k, e.g., we aim to maximize nDCG@k. In this case, the number of documents passed by the filter must have a similarity close enough to the similarity of the document at the k^{th} position in the ranked list. Let σ be this similarity and δ a *proximity* value with $\delta \geq 0$. Therefore, τ becomes equal to $\sigma - \delta$. For example, if $\sigma = 0.8$ and $\delta = 0.1$, we have that $\tau = 0.7$.

It is worth noting that the number of documents in L_q that pass the filter is generally not uniform, regardless of using EST or EPT. The only case in which the number of documents preserved by the filter is exactly k is when we adopt EPT with $\delta = 0$. Otherwise, if $\delta > 0$, besides these k documents, the further ones passed by the filter are those with a similarity in the range $[\sigma - \delta, \sigma)$.

In the remainder of the paper, we discuss experiments for SEE adopting either EST or EPT. As stated previously, we can choose the strictness of the filter of EST and EPT: we can directly vary τ for EST, whereas for EPT we indirectly tune τ by changing the value

of δ. For both techniques, another crucial ingredient is the *position* of the SEE filter, which we can place before any transformer blocks $T_0, \ldots, T_K$. Moreover, in principle, we can insert multiple early exit filters, thus forming a cascade. In this paper, we use a simpler—yet effective—approach that uses only a single filter. The earlier the filter is placed, the more conspicuous the potential reduction in latency will be.

4 Experimental Evaluation

In this section, we assess the efficiency and effectiveness of SEE. We first briefly describe our experimental setup. We then experimentally evaluate different similarity measures, filter placement alternatives, and choices of τ, the threshold that triggers SEE. Next, we present two in-depth analyses that compare the performance of 1) SEE applied to MonoBERT and eeMB [47] on a *in-domain* task on three distinct benchmark datasets, 2) SEE applied to MonoBERT and eeMB evaluated over multiple *out-of-domain* tasks using BEIR [40] datasets, and 3) SEE applied to MonoBERT and eeMB using different list truncation cutoffs. Finally, we highlight the adaptability of SEE by implementing and evaluating its performance when applied to different PLMs, namely MonoELECTRA, miniLM, and X-ELECTRA.

4.1 Experimental Setup

Following the seminal work of Nogueira et al. [30], we use two fine-tuned BERT on two datasets: MS MARCO [1] and a version of ASNQ [16] adapted for the ranking task [47][1]. In detail, for MS MARCO, we use the triples (query, positive document, and negative document) provided with the dataset. Instead, for ASNQ, we use the code provided by Xin *et al.* [47] to create the triples. In the same way, we also utilize MS MARCO to fine-tune a cross-encoder based on ELECTRA. We named it MonoELECTRA. Both MonoBERT and MonoELECTRA are composed of 12 layers each. In addition, we employ two other open-source models available within the HuggingFace [46] repository, with both being already fine-tuned on MS MARCO. In detail, we use miniLM [45] with 12 layers, available in the Sentence-transformer library [33], and a larger version of ELECTRA composed of 24 layers, which we call X-ELECTRA [11].

Evaluation. In-domain evaluation in MS MARCO is performed using the official (small) development query set and the 2019 TREC Deep Learning Track query set, hereinafter named *dev* and *trec-dl-2019*, respectively. For evaluating ASNQ, we generate our test dataset by processing the official development set in the same way as done for the training set, i.e., we only consider the relevant documents with a label equal to 4. In contrast, all the others were considered non-relevant. In addition, we evaluate our approach on all 15 publicly-available datasets in the BEIR [40] benchmark. We use the official candidate sets to be re-ranked provided by *dev*, *trec-dl-2019*, and ASNQ. In contrast, for BEIR, we compute a set of 1000 candidates for each query using the BM25 implementation available in the BM25s library [26], with $k_1 = 1.2$, and $b = 0.75$.

Competitors. As discussed in Sec. 2, to the best of our knowledge, eeMB [47] is the only method available for early stopping inference of MonoX models for ranking tasks. It relies on learned classifiers placed after each transformer block, which decide whether a query-document pair can be dropped or should proceed to the next transformer block. We fine-tune and evaluate eeMB on MS MARCO and ASNQ, following the authors' instructions and code. We use the proposed implementation as a baseline. Moreover, we provide an improved version, which we refer to as eeMB$_+$, which makes eeMB much faster in performing query-document inference in a real-world scenario, i.e., using a batch size greater than one. This new version revolves around a rebatching mechanism described in the following section.

Implementation Details. SEE exploits a rebatching mechanism after the filtering step. Rebatching improves the efficiency of the inference step on the GPU by grouping the non-early-exited documents before moving to the following transformer blocks. This allows for a more efficient use of GPU resources. The rebatching mechanism is query-wise, i.e., we rebatch the documents in L_q surviving the filter for a single query q. We implemented the same rebatching mechanism in eeMB. This is crucial to make fair comparisons between the two methods. This is because learned early-exit methods like eeMB involve unbalanced scoring costs, as query-document pairs scored for a query can traverse a different number of transformer blocks. This does not allow the full exploitation of GPU parallelism that enables query-document pairs to be scored in batches. It is worth noting that the paper presenting eeMB [47] lacks a detailed discussion on the implications of batch size during the inference phase. Thus, we propose a more efficient implementation of eeMB by splitting the model after every transformer block T_i and introducing a rebatching phase to improve the usage of GPU resources.

The efficiency tests of SEE and eeMB were performed on a single Nvidia A100-SXM4-40GB GPU.

4.2 Similarity Measures Evaluation

We start assessing whether similarity measures (Eq. 1 - Eq. 4) are a good proxy of the final scores produced by a cross-encoder. Therefore, regardless of the relevance labels in each dataset's ground truth, in this analysis, we evaluate the performance of the similarity functions by considering the ranking score produced by the whole model as a reference. Since the standard metrics used to assess the quality of a ranking define a cutoff point for the top-ranked items, e.g., nDCG@10, we assume that the top 10 documents obtained by the model are the relevant ones, and the rest of the ranked list is composed of non-relevant documents. We thus re-rank the set $L_q = [d_1, d_2, \ldots, d_{|L_q|}]$ of documents associated with a given query q by using $Sim(q, d_i)$, $d_i \in L_q$, where Sim is one of the four tested similarity functions and $|L_q|$ is usually equal to 1000. To assess whether embedding similarities are a good proxy of relevance, we measure how many of the top-10 scored documents returned by the model can be found in the top-100 documents in L_q after sorting the list by $Sim(d_i, q)$ in reverse order of similarity. This corresponds to measuring the *recall* at cutoff 100 (R@100) by considering the top-10 scored documents identified by the whole cross-encoder model as the only relevant ones. High values of R@100 guarantee that the top-10 scored documents by the full cross-encoder without EE can be found in the top-100 documents ranked by token embedding

[1]Code, links to models, additional data, and experimental settings are available here: https://github.com/veneres/SEE-SIGIR25

Table 1: R@100 achieved by four different similarity functions applied before T_0, T_4, T_8 and after T_{11}, marked respectively as 0, 4, 8, 12 on a sample of the training set of MS MARCO using MonoBERT. In this experiment, a document is considered relevant if present in one of the top-10 positions of the ranking produced by MonoBERT. The best result for each placement is underlined while the best overall is in bold.

Similarity	Filter position			
	0	4	8	12
Max	0.28	0.45	0.45	0.69
MeanSim	0.33	0.30	0.21	0.37
CentrSim	0.36	0.33	0.27	0.68
MaxSim	0.67	0.73	0.72	**0.85**

similarities for each query q, and thus the cross-encoder should be able to reconstruct the top-10 ranked documents computing the full score only for 100 instead of 1000. Note that the SEE filter used in the other experiments is more sophisticated, as the number of documents in L_q passed by the filter is dynamically chosen on a per-query basis.

Table 1 reports the average R@100 obtained on a sample of the official MS MARCO training set (with approximately 1000 candidates for each query) using MonoBERT. We chose to use a sample of the official training set as the validation set for the analysis of our filter because no official validation set exists in MS MARCO. Columns marked as 0, 4, 8, and 12 report the results obtained by applying $Sim(q, d_i)$, $d_i \in L_q$, to the embeddings before the transformer blocks T_0, T_4, T_8, and after T_{11}. The embeddings before T_0 are those produced by the model just after the tokenizer, i.e., the initial embeddings including sentence and positional components. In contrast, the embeddings after T_{11} (column 12) are the last ones produced by the transformers and thus are the inputs of the classifier layer of the PLM model. We observe that we obtain the best recall using the MaxSim similarity when using the embeddings after T_{11}. Moreover, with a R@100 above 0.67 achieved before T_0, we confirm that a similarity measure can be used to devise effective exit strategies in the early stage of a cross-encoder model.

These results confirm that if only the top-100 documents, sorted by embedding similarity using MaxSim, are passed by our SEE filter positioned before T_0, the final top-100 results will contain, on average, 7 of the top-10 results returned by the full MonoBERT without EE. In summary, the best results are obtained using MaxSim, and therefore, in the following experiments, we use MaxSim as the similarity measure implemented within our SEE.

4.3 Filter Placement Evaluation

Another key ingredient for the proposed method is the choice of the EE filter placement. In Fig. 2, we show the efficiency/effectiveness trade-offs of different filter positions using MonoBERT on the same training set subset used in the previous section. We report the *estimated speedup* (SU_e), which, similarly to Xin *et al.* [47], is computed as the ratio between the number of transformer blocks traversed with and without SSE. This is linked to the estimated ratio of floating point operations performed with and without the EE strategy.

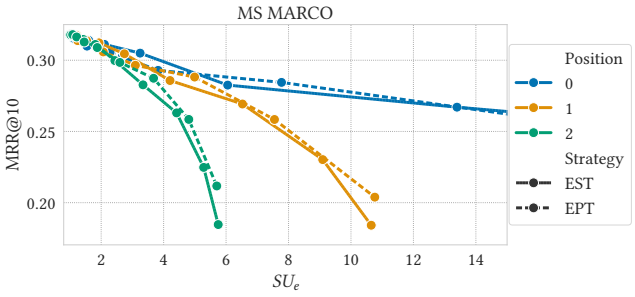


Figure 2: Comparison on a *training set sample* of MS MARCO dataset using MonoBERT in terms of MRR@10 and *estimated* speedup SU_e using different filter positions and SEE strategies. For our two strategies, EPT and EST, each point represents a different value of τ.

To reduce cluttering, we report only a subset of all possible choices of filter placement; in particular, we show the effect of a filter before T_0, T_1, and T_2. Each point of the lines represents a different threshold choice that ranges from 0.1 to 0.9, with a step of 0.1, for both EST and EPT. From the figure, we can highlight two main results. First, we observe that EPT consistently outperforms EST, regardless of the filter position. Second, by placing the filter before T_0, we obtain better trade-offs with both strategies. Therefore, in the following experiments, we place SEE before T_0, and we use only EPT, as this combination provides a better (or at least equal) efficiency/effectiveness trade-off than all other possible choices explored.

4.4 Performance Evaluation

In this section, we report a comprehensive performance evaluation of SEE using MonoBERT and 16 datasets, i.e., *dev*, *trec-dl-2019*, *ASNQ*, and the 14 datasets of the BEIR benchmark.[2] The goal of the proposed evaluation is to compare SEE and eeMB$_+$, i.e., our—improved—implementation of eeMB that employs rebatching to improve the utilization of GPU resources during inference. Following the results in Table 1 and Fig. 2 we summarize our experimental choices as follows: SEE uses MaxSim, and we employ both SEE and EPT before T_0. Instead, for eeMB, we base our choices on the results proposed by Xin et al. [47], which means that we fix the positive confidence threshold to 1.00 since it gives the best efficiency/effectiveness trade-off, varying only the negative confidence c_n.

For a fair comparison, we first present an evaluation based on estimated/actual speed-ups for eeMB, eeMB$_+$, and SEE using the same configurations and datasets presented in the original paper of eeMB. Then, we present an evaluation of the two approaches on the 14 out-of-domain datasets of the BEIR benchmark to assess the generalizability of our method to different corpus. To further assess the applicability of our method to different realistic scenarios, we present the results of the comparison between SEE and eeMB$_+$

[2]Indeed, BEIR consists of a total of 15 datasets; however, one of them, MS MARCO (trec-dl-2019), is also analyzed separately.

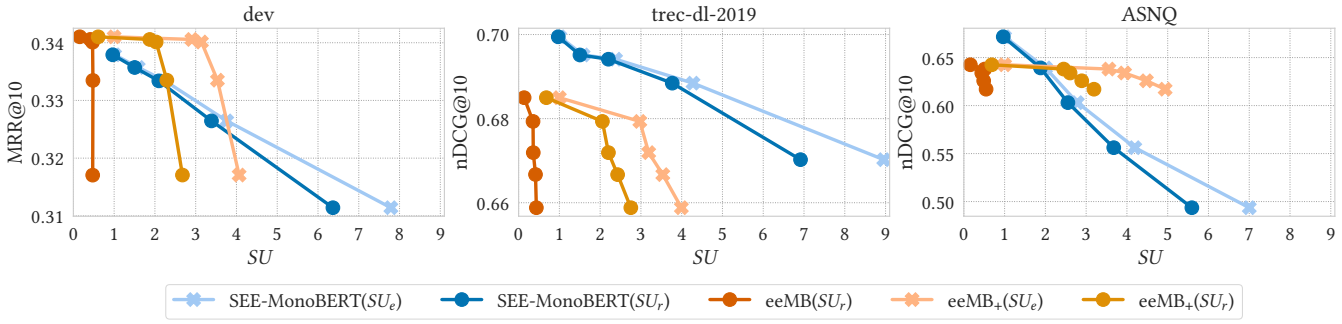


Figure 3: Comparison in terms of effectiveness and *estimated* speedup SU_e and real speedup SU_r between SEE applied on MonoBERT and eeMB. For SEE, each point represents a different threshold value τ, while for eeMB, a different value of c_n. We have increasingly stricter filter thresholds for all methods from left to right.

when combined with a ranked list truncation strategy to limit the number of documents to be re-ranked.

In-domain Evaluation. We first focus on Fig. 3 showing the performance of SEE applied to MonoBERT (denoted by SEE-MonoBERT) and eeMB on three different datasets. For the experiments reported in the figure, the negative confidence c_n of eeMB and eeMB$_+$ was set in the range $c_n \in \{0.7, 0.8, 0.9, 0.95, 1\}$, while for SEE-MonoBERT, the parameter δ of EPT was set in the range $\delta \in \{0.2, 0.3, 0.4, 0.5, 0.9\}$. Given the importance of the implementation details and the real speedup of the proposed strategy, we report both the curve of the estimated and real speedup (SU_e and SU_r). We report both the results of our version of eeMB, denoted by eeMB$_+$, enhanced with a novel rebatching mechanism to exploit modern high-end GPUs, and the original eeMB implementation.

Focusing on Fig. 3, each curve represents the performance of a different EE technique for MonoBERT on a given dataset. In particular, any point of each curve shows the effectiveness (mRR@10 or nDCG@10) and the speedup (SU_e or SU_r) obtained by an EE technique for a given configuration of its parameters (either τ or c_n). All the speedups are computed against MonoBERT without EE, corresponding to the point with SU=1 of the curve SEE-MonoBERT. For each curve, from left to right the parameter setting increases the amount of document pairs that are early exited, thus producing larger speedups with a small loss in effectiveness. As can be seen in Fig. 3, eeMB never achieves a real speedup (SU_r) greater than 1. Thus, in the next paragraphs, we only focus on eeMB$_+$, that by integrating our re-batching mechanism is much more efficient and closer to the estimated speedup. We highlight that the versions of MonoBERT and eeMB$_+$ used for the evaluation on ASNQ and MS MARCO (*dev* and *trec-dl-2019*) were fine-tuned on ASNQ and MS MARCO respectively, and thus in this experiment we show only an in-domain evaluation.

The figure shows that the effectiveness of SEE is always higher on *trec-dl-2019*, maintaining a higher nDCG@10 than eeMB$_+$ regardless of the speedup (more than 3.5× times faster with a decrease of less than 2%). We highlight that *trec-dl-2019* is the only dataset considered in this comparison with more than one relevance judgment.

Conversely, for *dev*, we observe that monoBERT has a base lower effectiveness than eeMB$_+$, i.e., when the EE is not performed, and the speedup is 1.0. This fact also entails that SEE outperforms, in our real-timing evaluation experiments, the classifier-based approach only when the SU_r is above 2.5. However, we still observe a lower steepness of the curve, which gives a more predictable behavior of SEE when changing the exit parameters. We also observe that in ASNQ, a MonoBERT without EE outperforms a eeMB$_+$ with a negative threshold c_n very high, i.e., $n_c = 1$, and thus in practice without EE. However, SEE has a decisive decrease in effectiveness as we make our SEE filter weaker to increase speedup and quickly becomes worse than eeMB$_+$.

Notably, Fig. 3 shows the difference between the estimated speedup and the real speedup using SEE and eeMB$_+$. In detail we remark that using SEE the difference between estimated and measured latency, i.e., SU_e and SU_r, respectively, becomes large only when SU_e is greater than 4×, while in eeMB$_+$ this is always significant. This phenomenon can be explained by observing that eeMB$_+$ requires many re-batching steps (one every layer). Each of them adds a computational overhead to the approach. In contrast, SEE has only one re-batching step after the only exit decision is performed.

Out-of-domain Evaluation. We now present an out-of-domain evaluation by comparing SEE on MonoBERT and eeMB$_+$ on the BEIR datasets. In this analysis, we employ MonoBERT and eeMB, both fine-tuned on MS MARCO, aiming to evaluate the ability of SEE to cope with the zero-shot usage of cross-encoders.

The results of the study are presented in Table 2 where we report the performance obtained in the 15 BEIR datasets using MonoBERT using or not the two EE strategies under investigation. The datasets presented in Table 2 are grouped by their task. Going from the top to the bottom, we have datasets for passage retrieval, bio-medical information retrieval, question answering, argument retrieval, duplicate-question retrieval, entity retrieval, citation prediction, and fact-checking. At the bottom, we report the row containing the average nDCG@10 and the query latency (in seconds) of each method analyzed. In this part of the analysis, we focus only on the results obtained using a candidate list of 1000 documents (rightmost group of columns in Table 2). In these experiments, given the number of datasets and thus query-document pairs to be re-ranked, we fixed the parameters for SEE and eeMB instead of conducting a fine-grained evaluation of both techniques.

Table 2: Assessment of out-of-domain, except for MS MARCO (trec-dl-2019), SEE performance on the BEIR benchmark. For each model and dataset, we report the nDCG@10 in percentage and, between parenthesis, the average per-query inference time (in seconds) or the relative speedup. We also differentiate the results obtained by truncating the BM25-ranked list to different sizes. SEE and eeMB are compared with the performance obtained when no EE strategy is applied in a standard MonoBERT. Statistically significant improvements are marked with ↑, while effectiveness degradations are marked with ↓. For each dataset and truncation list size, we highlight in bold the best efficiency/effectiveness tradeoff obtained.

BM25 List Size →	100			500			1000		
Dataset ↓	MonoBERT	eeMB$_+$ ($c_n = 0.95$)	SEE ($\delta = 0.3$)	MonoBERT	eeMB$_+$ ($c_n = 0.95$)	SEE ($\delta = 0.3$)	MonoBERT	eeMB$_+$ ($c_n = 0.95$)	SEE ($\delta = 0.3$)
MS MARCO [1]	69.9 (0.23)	68.0 (1.3×)	**66.6 (1.7×)**	68.7 (1.08)	68.2 (1.8×)	**66.1 (2.7×)**	68.1 (2.12)	68.1 (2.0×)	**67.1 (3.1×)**
TREC-COVID [42]	66.0 (0.22)	64.9 (0.9×)	**65.9 (1.1×)**	61.5 (1.07)	62.9 (0.9×)	**61.4 (1.2×)**	61.5 (2.13)	59.9 (0.9×)	**61.5 (1.2×)**
NFCorpus [3]	34.1 (0.22)	↓30.2 (2.4×)	**33.4 (1.5×)**	34.1 (1.06)	↓29.0 (2.7×)	**33.5 (2.6×)**	34.0 (2.11)	↓28.5 (2.6×)	**33.6 (3.5×)**
Natural Questions [23]	45.5 (0.21)	**44.8 (1.5×)**	↓44.3 (1.4×)	47.1 (1.06)	**46.4 (2.0×)**	↓45.7 (1.7×)	47.4 (2.12)	**46.7 (2.1×)**	↓46.1 (1.8×)
HotpotQA [51]	70.3 (0.21)	**↓69.3 (1.6×)**	↓68.9 (1.4×)	71.0 (1.06)	**↓70.0 (2.0×)**	↓69.7 (1.8×)	71.2 (2.11)	**↓70.1 (2.0×)**	↓69.9 (1.9×)
FIQA [28]	36.0 (0.21)	**↓32.7 (1.5×)**	↓32.5 (1.6×)	35.9 (1.06)	↓32.2 (2.0×)	**↓32.8 (2.2×)**	36.2 (2.11)	↓32.4 (2.1×)	**↓33.1 (2.5×)**
ArguANA[43]	39.8 (0.21)	↓33.2 (1.7×)	**39.3 (1.0×)**	39.9 (1.06)	↓32.5 (2.3×)	**↓39.2 (1.1×)**	39.8 (2.13)	↓32.4 (2.3×)	**39.1 (1.1×)**
Touché-2020 [2]	31.8 (0.23)	33.3 (1.1×)	**32.7 (1.2×)**	30.6 (1.07)	32.6 (1.4×)	**31.5 (1.4×)**	30.4 (2.13)	**32.6 (1.6×)**	31.0 (1.5×)
CQADupStack [21]	33.5 (0.21)	↓29.7 (1.5×)	**↓32.3 (1.5×)**	33.5 (1.06)	↓29.2 (1.9×)	**↓32.4 (1.9×)**	33.5 (2.11)	↓29.1 (2.0×)	**↓32.5 (2.2×)**
Quora	82.9 (0.21)	↓68.7 (1.8×)	**↓81.1 (1.5×)**	82.3 (1.06)	↓67.4 (2.4×)	**↓80.7 (1.9×)**	82.2 (2.11)	↓67.1 (2.6×)	**↓80.7 (2.2×)**
DBPedia [18]	39.9 (0.21)	**39.0 (1.4×)**	↓38.5 (1.6×)	39.9 (1.06)	39.5 (1.9×)	**39.7 (2.4×)**	39.8 (2.11)	39.3 (2.0×)	**39.6 (2.8×)**
SCIDOCS [9]	14.8 (0.21)	↓13.4 (1.4×)	**14.6 (1.4×)**	14.9 (1.06)	↓13.1 (1.9×)	**14.7 (1.5×)**	14.8 (2.11)	↓13.1 (2.1×)	**14.7 (1.5×)**
FEVER [41]	73.8 (0.21)	73.7 (1.5×)	**↑74.1 (1.5×)**	75.6 (1.06)	**75.5 (1.9×)**	↑76.1 (1.9×)	76.0 (2.11)	**75.9 (2.0×)**	↑76.5 (2.0×)
SciFact [44]	69.1 (0.21)	↓64.9 (1.8×)	**68.6 (1.4×)**	69.7 (1.06)	↓64.7 (2.4×)	**69.5 (2.2×)**	69.7 (2.12)	↓64.5 (2.4×)	**69.4 (2.7×)**
Climate-FEVER [24]	22.5 (0.21)	↓20.5 (2.1×)	**22.5 (1.5×)**	22.5 (1.06)	↓20.1 (2.5×)	**23.0 (1.8×)**	22.4 (2.13)	↓19.9 (2.5×)	**↑23.0 (1.9×)**
Average	48.7 (0.22)	45.8 (1.5×)	47.7 (1.4×)	48.5 (1.06)	45.6 (1.9×)	47.7 (1.8×)	48.5 (2.12)	45.3 (1.9×)	47.9 (1.9×)

We recall that for SEE, we use EPT with $\delta = 0.3$, as it shows a good efficiency/effectiveness tradeoff among all the three datasets in Fig. 3. In contrast, we fixed the negative confidence c_n of eeMB to 0.95, which is the threshold where we start to observe a steep degradation of effectiveness for eeMB, as depicted in Fig. 3. We evaluated the statistical significance of the nDCG differences between EE and non-EE model versions using a paired t-test ($p = 0.01$), applying Bonferroni correction for multiple comparisons [14]. The EE strategy with the best efficiency/effectiveness trade-off for MonoBERT is defined as the one with the highest speedup if no statistical difference is reported between the average nDCG@10 of the EE and no-EE versions. If both SEE and eeMB$_+$ exhibit statistically significant degradations, we highlight the one with the highest nDCG@10.

Looking at Table 2, we can clearly see that SEE presents a better efficiency/effectiveness tradeoff for the majority of datasets. In particular, SEE outperforms eeMB$_+$ on 11 out of 15 datasets. We also highlight that SEE applied to MonoBERT has a statistically significant effectiveness degradation only in 5 datasets, i.e., for two specific tasks: question answering and duplicate-question retrieval, while eeMB$_+$ shows significant drops in 8 datasets. This confirms that our SEE works well in an out-of-domain setting, reaching speedups of up to 3.5× without any learning involved.

Conversely, eeMB$_+$ struggles to maintain the same level of effectiveness as the MonoBERT without EE, also with a very conservative configuration. Overall, SEE achieves a speedup of 1.9× by losing 1.24% of nDCG@10, while eeMB$_+$ achieves the same speedup by losing 6.60% of nDCG@10.

Early Exit and Ranked List Truncation. We conclude the comparison between SEE and eeMB$_+$ by assessing the two methods using different candidate list sizes retrieved by BM25. This simulates the application of a RLT strategy aiming at reducing the number of documents to be re-ranked from the "standard" rank list size of 1000 documents. In particular, we used one popular RLT strategy that cuts the retrieved document list to a fixed cutoff [29].

In addition to the original size of 1000 we further consider sizes of 500 and 100. The results, presented in Table 2, confirm the better efficiency/effectiveness trade-off of SEE compared to eeMB$_+$ on most datasets, regardless of the list size used. In particular, eeMB$_+$ outperforms SEE only in 4 and 3 datasets out of 15 for cutoffs of 100 and 500, respectively. Finally, we highlight that SEE is still beneficial even when applied to a short candidate list. For instance, with a size of 100 SEE achieves an average speed-up of about 1.4× (from 220msec. to 150msec.) by loosing only 2% of nDCG@10.

Evaluation Summary We experimentally confirm that SEE can be a valid—non-learned—method to perform EE with a pointwise ranker as MonoBERT. While it shows comparable performance with other EE techniques, i.e., eeMB$_+$, in an in-domain setting, it notably outperforms them on multiple out-of-domain datasets. Moreover, it is an effective solution when combined with RLT for all sizes of candidate lists evaluated, especially when the number of elements to be re-ranked decreases significantly.

4.5 SEE Adaptability

To assess the adaptability of SEE, we implemented SEE on three cross-encoders besides MonoBERT: MonoELECTRA, miniLM, and

Table 3: SEE adaptability on BEIR with three cross-encoders: for each model and dataset, we report nDCG@10 (%) and average per-query inference time (s) with speedup (in parentheses). SEE is compared to the no-EE baseline. Statistically significant gains and drops are marked with ↑ and ↓, respectively; bold indicates speedups without significant effectiveness loss. .

Model →	monoELECTRA		miniLM		X-ELECTRA	
Dataset ↓	no EE	SEE ($\delta = 0.3$)	no EE	SEE ($\delta = 0.3$)	no EE	SEE ($\delta = 0.3$)
MS MARCO [1]	70.0 (2.10)	**68.2 (0.69/3.0×)**	75.6 (1.37)	**72.3 (0.46/3.0×)**	75.7 (6.00)	↓71.4 (1.85/3.3×)
TREC-COVID [42]	58.8 (2.08)	**59.0 (1.76/1.2×)**	63.8 (1.36)	**63.8 (1.15/1.2×)**	76.6 (5.98)	**76.6 (4.79/1.2×)**
NFCorpus [3]	26.7 (2.09)	**↑31.0 (0.58/3.6×)**	28.8 (1.36)	**↑30.7 (0.40/3.4×)**	35.4 (6.00)	**34.8 (1.60/3.8×)**
Natural Questions [23]	47.8 (2.36)	↓46.5 (1.38/1.7×)	49.6 (1.58)	↓48.2 (0.91/1.7×)	57.1 (6.48)	↓53.7 (3.13/2.1×)
HotpotQA [51]	61.8 (2.07)	↓60.7 (1.08/1.9×)	73.2 (1.35)	↓71.2 (0.71/1.9×)	75.9 (5.96)	↓73.5 (2.91/2.1×)
FIQA [28]	38.0 (2.08)	↓35.0 (0.81/2.6×)	36.8 (1.36)	↓33.1 (0.53/2.6×)	48.7 (5.97)	↓42.0 (2.26/2.6×)
ArguANA[43]	9.0 (2.08)	**8.9 (1.87/1.1×)**	20.5 (1.36)	**20.5 (1.17/1.2×)**	10.6 (5.97)	**10.3 (5.32/1.1×)**
Touché-2020 [2]	29.4 (2.11)	**30.4 (1.28/1.6×)**	34.8 (1.38)	**35.2 (0.87/1.6×)**	37.6 (6.03)	**38.1 (3.38/1.8×)**
CQADupStack [21]	32.7 (2.09)	↓31.8 (0.97/2.1×)	33.2 (1.36)	↓32.3 (0.64/2.1×)	38.6 (6.01)	↓36.4 (2.68/2.2×)
Quora	80.9 (2.08)	↓79.7 (0.93/2.2×)	82.1 (1.38)	↓80.2 (0.60/2.3×)	81.2 (5.97)	↓79.7 (2.58/2.3×)
DBPedia [18]	39.8 (2.08)	**40.1 (0.76/2.7×)**	45.1 (1.36)	↓43.8 (0.51/2.6×)	46.8 (5.97)	↓45.5 (2.07/2.9×)
SCIDOCS [9]	14.6 (2.08)	**14.4 (1.46/1.4×)**	15.3 (1.36)	**15.2 (0.97/1.4×)**	18.0 (5.97)	**17.8 (3.81/1.6×)**
FEVER [41]	74.1 (2.15)	**↑74.4 (1.09/2.0×)**	80.4 (1.43)	↓80.1 (0.75/1.9×)	82.8 (5.97)	↓82.5 (2.97/2.0×)
SciFact [44]	57.3 (2.08)	**↑59.7 (0.88/2.4×)**	66.8 (1.36)	**66.6 (0.53/2.5×)**	72.4 (5.98)	**72.5 (2.15/2.8×)**
Climate-FEVER [24]	11.6 (2.08)	**↑12.1 (1.16/1.8×)**	24.5 (1.36)	**↑25.0 (0.73/1.9×)**	21.0 (5.97)	**21.3 (3.08/1.9×)**
Average	43.5 (2.11)	43.5 (1.11/1.9×)	48.7 (1.38)	47.9 (0.73/1.9×)	51.9 (6.02)	50.4 (2.97/2.0×)

X-ELECTRA. As detailed in Section 4.1, the former is a MonoX model based on ELECTRA fine-tuned by us, the second is a strong but lightweight cross-encoder available in the *sentence-transformer* library [33], and the latter is a strong cross-encoder based on ELECTRA large [11]. The results, shown in Table 3, follow the evaluation strategy of Table 2, using the same parameters and statistical tests with a fixed candidate list of 1000 documents.

The results in Table 3 confirm that our method is effective across all three cross-encoders, achieving an average 2× speedup with only a limited drop in nDCG@10. While there is a statistically significant reduction in effectiveness on some datasets, 5 out of 15 for MonoELECTRA, 7 for miniLM, and 8 for X-ELECTRA, the overall performance remains competitive. SEE allows the use of bigger and more effective models without sacrificing ranking effectiveness. For instance, on NFCorpus, instead of employing MonoELECTRA that achieves 26.7% of nDCG@10 with an average query latency of more than 2 seconds per query, SEE enables the use of X-ELECTRA, achieving a better nDCG@10 (34.8%) while lowering the average query latency to 1.60 seconds. The same choice can also be applied to DBPedia where, with the same average query latency, the improvement in terms of nDCG@10 from MonoELECTRA without EE to X-ELECTRA with SEE is about 17%. As a side note, we highlight the poor effectiveness of MonoELECTRA, miniLM, and X-ELECTRA on ArguANA. From a preliminary analysis, we found that the problem is due to the length of the ArguANA queries, i.e., 192 words in average.

To summarize, since SEE does not require additional training, we show that it can be easily adapted to different cross-encoders. In this setting, we show that SEE allows for a speedup of up to 3.8× with a limited loss in ranking effectiveness.

5 Conclusion and Future Work

We presented SEE, a novel early-exit strategy for cross-encoder models used in document re-ranking. First, we showed that the similarities of the embeddings during the inference phase can be successfully exploited to predict the final ranking of the model. Second, we introduced a principled early-exit strategy that exploits this intuition. Moreover, it does not require additional learning on top of the cross-encoder models. We performed an extensive analysis of SEE on 17 public datasets and models and we showed how SEE is able to achieve speedups of up to 3.5× with a limited loss of nDCG@10. As future work, we plan to improve the real speedup of SEE by further optimizing its implementation on GPU, while we would also like to extend our work by using multiple SEE filters and study their effectiveness/efficiency tradeoffs.

Acknowledgements This work was partially supported by the EU project EFRA (Grant 101093026), and by the following Next Generation EU (EU-NGEU) projects: SERICS (Grant NRRP M4C2 Inv.1.3 PE00000014), and FAIR (Spoke 1, Grant NRRP M4C2 Inv.1.3 PE00000013). The views and opinions expressed are solely those of the authors and do not necessarily reflect those of the European Union, nor can the European Union be held responsible for them.

References

[1] Payal Bajaj et al. 2018. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. arXiv:1611.09268 [cs]. (Oct. 2018). Retrieved Oct. 6, 2022 from http://arxiv.org/abs/1611.09268.

[2] Alexander Bondarenko et al. 2020. Overview of touché 2020: argument retrieval. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 11th International Conference of the CLEF Association, CLEF 2020, Thessaloniki, Greece, September 22–25, 2020, Proceedings 11*. Springer, 384–395.

[3] Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A Full-Text Learning to Rank Dataset for Medical Information Retrieval. en. In *Advances in Information Retrieval*. Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Giorgio Maria Di Nunzio, Claudia Hauff, and Gianmaria Silvello, (Eds.) Springer International Publishing, Cham, 716–722. ISBN: 978-3-319-30671-1. doi:10.1007/978-3-319-30671-1_58.

[4] Sebastian Bruch, Claudio Lucchese, and Franco Maria Nardini. 2023. Efficient and effective tree-based and neural learning to rank. *Foundations and Trends® in Information Retrieval*, 17, 1, 1–123. doi:10.1561/1500000071.

[5] Sebastian Bruch, Claudio Lucchese, and Franco Maria Nardini. 2023. Efficient and effective tree-based and neural learning to rank. *Found. Trends Inf. Retr.*, 17, 1, 1–123. doi:10.1561/1500000071.

[6] Francesco Busolin, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. 2023. Early exit strategies for learning-to-rank cascades. *IEEE Access*, 11, 126691–126704. doi:10.1109/ACCESS.2023.3331088.

[7] Francesco Busolin, Claudio Lucchese, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Salvatore Trani. 2021. Learning early exit strategies for additive ranking ensembles. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*. Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai, (Eds.) ACM, 2217–2221. doi:10.1145/3404835.3463088.

[8] B. Barla Cambazoglu, Hugo Zaragoza, Olivier Chapelle, Jiang Chen, Ciya Liao, Zhaohui Zheng, and Jon Degenhardt. 2010. Early exit optimizations for additive machine learned ranking systems. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining* (WSDM '10). Association for Computing Machinery, New York, New York, USA, 411–420. ISBN: 9781605588896. doi:10.1145/1718487.1718538.

[9] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. 2020. SPECTER: Document-level Representation Learning using Citation-informed Transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, (Eds.) Association for Computational Linguistics, Online, (July 2020), 2270–2282. doi:10.18653/v1/2020.acl-main.207.

[10] Claudio M. V. de Andrade, Fabiano M. Belém, Washington Cunha, Celso França, Felipe Viegas, Leonardo Rocha, and Marcos André Gonçalves. 2023. On the class separability of contextual embeddings representations – or "The classifier does not matter when the (text) representation is so good!". *Information Processing & Management*, 60, 4, (July 2023), 103336. doi:10.1016/j.ipm.2023.103336.

[11] Hervé Déjean, Stéphane Clinchant, and Thibault Formal. 2024. A thorough comparison of cross-encoders and llms for reranking splade. *arXiv preprint arXiv:2403.10407*.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Jill Burstein, Christy Doran, and Thamar Solorio, (Eds.) Association for Computational Linguistics, Minneapolis, Minnesota, (June 2019), 4171–4186. https://aclanthology.org/N19-1423.

[13] Yixing Fan, Xiaohui Xie, Yinqiong Cai, Jia Chen, Xinyu Ma, Xiangsheng Li, Ruqing Zhang, and Jiafeng Guo. 2022. Pre-training Methods in Information Retrieval. English. *Foundations and Trends® in Information Retrieval*, 16, 3, (Aug. 2022), 178–317. Publisher: Now Publishers, Inc. doi:10.1561/1500000100.

[14] Norbert Fuhr. 2018. Some common mistakes in ir evaluation, and how they can be avoided. *SIGIR Forum*, 51, 3, (Feb. 2018), 32–41. doi:10.1145/3190580.3190586.

[15] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2020. Understanding bert rankers under distillation. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval* (ICTIR '20). Association for Computing Machinery, Virtual Event, Norway, 149–152. ISBN: 9781450380676. doi:10.1145/3409256.3409838.

[16] Siddhant Garg, Thuy Vu, and Alessandro Moschitti. 2020. Tanda: transfer and adapt pre-trained transformer models for answer sentence selection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34, 05, (Apr. 2020), 7780–7788. doi:10.1609/aaai.v34i05.6282.

[17] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W. Bruce Croft, and Xueqi Cheng. 2020. A Deep Look into neural ranking models for information retrieval. *Information Processing & Management*, 57, 6, (Nov. 2020), 102067. doi:10.1016/j.ipm.2019.102067.

[18] Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. DBpedia-Entity v2: A Test Collection for Entity Search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '17). Association for Computing Machinery, New York, NY, USA, (Aug. 2017), 1265–1268. ISBN: 978-1-4503-5022-8. doi:10.1145/3077136.3080751.

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network. arXiv:1503.02531 [cs, stat]. (Mar. 2015). doi:10.48550/arXiv.1503.02531.

[20] Sebastian Hofstätter, Bhaskar Mitra, Hamed Zamani, Nick Craswell, and Allan Hanbury. 2021. Intra-document cascading: learning to select passages for neural document ranking. In (SIGIR '21). Association for Computing Machinery, , Virtual Event, Canada, 1349–1358. ISBN: 9781450380379. doi:10.1145/3404835.3462889.

[21] Doris Hoogeveen, Karin M. Verspoor, and Timothy Baldwin. 2015. CQADupStack: A Benchmark Data Set for Community Question-Answering Research. In *Proceedings of the 20th Australasian Document Computing Symposium* (ADCS '15). Association for Computing Machinery, New York, NY, USA, (Dec. 2015), 1–8. ISBN: 978-1-4503-4040-3. doi:10.1145/2838931.2838934.

[22] Omar Khattab and Matei Zaharia. 2020. Colbert: efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '20). Association for Computing Machinery, Virtual Event, China, 39–48. ISBN: 9781450380164. doi:10.1145/3397271.3401075.

[23] Tom Kwiatkowski et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

[24] Markus Leippold and Thomas Diggelmann. 2020. Climate-FEVER: A Dataset for Verification of Real-World Climate Claims. en-US. In *Climate Change AI*. Climate Change AI, (Dec. 2020). Retrieved Aug. 11, 2024 from https://www.climatechange.ai/papers/neurips2020/67.

[25] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2022. *Pretrained Transformers for Text Ranking: BERT and Beyond*. en. *Synthesis Lectures on Human Language Technologies*. Springer International Publishing, Cham. ISBN: 978-3-031-01053-8. doi:10.1007/978-3-031-02181-7.

[26] Xing Han Lù. 2024. Bm25s: orders of magnitude faster lexical search via eager sparse scoring. (2024). https://arxiv.org/abs/2407.03618 arXiv: 2407.03618 [cs.IR].

[27] Claudio Lucchese, Giorgia Minello, Franco Maria Nardini, Salvatore Orlando, Raffaele Perego, and Alberto Veneri. 2023. Can Embeddings Analysis Explain Large Language Model Ranking? In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management* (CIKM '23). Association for Computing Machinery, New York, NY, USA, (Oct. 2023), 4150–4154. ISBN: 9798400701245. doi:10.1145/3583780.3615225.

[28] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. WWW'18 Open Challenge: Financial Opinion Mining and Question Answering. In *Companion Proceedings of the The Web Conference 2018* (WWW '18). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, (Apr. 2018), 1941–1942. ISBN: 978-1-4503-5640-4. doi:10.1145/3184558.3192301.

[29] Chuan Meng, Negar Arabzadeh, Arian Askari, Mohammad Aliannejadi, and Maarten de Rijke. 2024. Ranked List Truncation for Large Language Model-based Re-Ranking. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '24). Association for Computing Machinery, New York, NY, USA, (July 2024), 141–151. ISBN: 9798400704314. doi:10.1145/3626772.3657864.

[30] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-Stage Document Ranking with BERT. arXiv:1910.14424 [cs]. (Oct. 2019). doi:10.48550/arXiv.1910.14424.

[31] Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. en. In (Feb. 2018). Retrieved Jan. 15, 2024 from https://openreview.net/forum?id=S1XolQbRW.

[32] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! arXiv:2312.02724 [cs]. (Dec. 2023). doi:10.48550/arXiv.2312.02724.

[33] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, (Eds.) Association for Computational Linguistics, Hong Kong, China, (Nov. 2019), 3982–3992. doi:10.18653/v1/D19-1410.

[34] Lior Rokach and Oded Maimon. 2005. Clustering Methods. In *Data Mining and Knowledge Discovery Handbook*. Oded Maimon and Lior Rokach, (Eds.) Springer US, Boston, MA, 321–352. ISBN: 978-0-387-25465-4. doi:10.1007/0-387-25465-X_15.

[35] Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022. Plaid: an efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (CIKM '22). Association for Computing Machinery, Atlanta, GA, USA, 1747–1756. ISBN: 9781450392365. doi:10.1145/3511808.3557325.

[36] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, (Eds.) Association for Computational Linguistics, Seattle, United States, (July 2022), 3715–3734. doi:10.18653/v1/2022.naacl-main.272.

[37] Harrisen Scells, Shengyao Zhuang, and Guido Zuccon. 2022. Reduce, reuse, recycle: green information retrieval research. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '22). Association for Computing Machinery, New York, NY, USA, 2825–2837. ISBN: 9781450387323. doi:10.1145/3477495.3531766.

[38] Ferdinand Schlatt, Maik Fröbe, Harrisen Scells, Shengyao Zhuang, Bevan Koopman, Guido Zuccon, Benno Stein, Martin Potthast, and Matthias Hagen. 2024. Set-Encoder: Permutation-Invariant Inter-Passage Attention for Listwise Passage Re-Ranking with Cross-Encoders. arXiv:2404.06912 [cs]. (June 2024). doi:10.48550/arXiv.2404.06912.

[39] Manveer Singh Tamber, Ronak Pradeep, and Jimmy Lin. 2023. Scaling Down, LiTting Up: Efficient Zero-Shot Listwise Reranking with Seq2seq Encoder-Decoder Models. arXiv:2312.16098 [cs]. (Dec. 2023). doi:10.48550/arXiv.2312.16098.

[40] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. en. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 1, (Dec. 2021). Retrieved Aug. 11, 2024 from https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/65b9eea6e1cc6bb9f0cd2a47751a186f-Abstract-round2.html.

[41] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Marilyn Walker, Heng Ji, and Amanda Stent, (Eds.) Association for Computational Linguistics, New Orleans, Louisiana, (June 2018), 809–819. doi:10.18653/v1/N18-1074.

[42] Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2021. TREC-COVID: constructing a pandemic information retrieval test collection. *SIGIR Forum*, 54, 1, (Feb. 2021), 1:1–1:12. doi:10.1145/3451964.3451965.

[43] Henning Wachsmuth, Shahbaz Syed, and Benno Stein. 2018. Retrieval of the Best Counterargument without Prior Topic Knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Iryna Gurevych and Yusuke Miyao, (Eds.) Association for Computational Linguistics, Melbourne, Australia, (July 2018), 241–251. doi:10.18653/v1/P18-1023.

[44] David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or Fiction: Verifying Scientific Claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, (Eds.) Association for Computational Linguistics, Online, (Nov. 2020), 7534–7550. doi:10.18653/v1/2020.emnlp-main.609.

[45] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. In *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 5776–5788. Retrieved Aug. 11, 2024 from https://proceedings.neurips.cc/paper/2020/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[46] Thomas Wolf et al. 2019. Huggingface's transformers: state-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

[47] Ji Xin, Rodrigo Nogueira, Yaoliang Yu, and Jimmy Lin. 2020. Early exiting BERT for efficient document ranking. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*. Nafise Sadat Moosavi, Angela Fan, Vered Shwartz, Goran Glavaš, Shafiq Joty, Alex Wang, and Thomas Wolf, (Eds.) Association for Computational Linguistics, Online, (Nov. 2020), 83–88. doi:10.18653/v1/2020.sustainlp-1.11.

[48] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, (Eds.) Association for Computational Linguistics, Online, (July 2020), 2246–2251. doi:10.18653/v1/2020.acl-main.204.

[49] Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. BERxiT: early exiting for BERT with better fine-tuning and extension to regression. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty, (Eds.) Association for Computational Linguistics, Online, (Apr. 2021), 91–104. doi:10.18653/v1/2021.eacl-main.8.

[50] Canwen Xu and Julian McAuley. 2023. A Survey on Model Compression and Acceleration for Pretrained Language Models. en. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37, 9, (June 2023), 10566–10575. doi:10.1609/aaai.v37i9.26255.

[51] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, (Eds.) Association for Computational Linguistics, Brussels, Belgium, (Oct. 2018), 2369–2380. doi:10.18653/v1/D18-1259.

[52] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33, 18330–18341.