ED2LM: Encoder-Decoder to Language Model for Faster Document Re-ranking Inference

Anonymous ACL submission

Abstract

State-of-the-art neural models typically encode document-query pairs using cross-attention for re-ranking. To this end, models generally utilize an encoder-only (like BERT) paradigm or an encoder-decoder (like T5) approach. These paradigms, however, are not without flaws, i.e., running the model on all query-document pairs at inference-time incurs a significant computational cost. This paper proposes a new training and inference paradigm for re-ranking. We propose to finetune a pretrained encoder-decoder model using in the form of document to query generation. Subsequently, we show that this encoder-decoder architecture can be decomposed into a decoder-only language model during inference. This results in significant inference time speedups since the decoder-only architecture only needs to learn to interpret static encoder embeddings during inference. Our experiments show that this new paradigm achieves results that are comparable to the more expensive cross-attention ranking approaches while being up to 6.8X faster. We believe this work paves the way for more efficient neural rankers that leverage large pretrained models.

1 Introduction

Leveraging transformer architecture to model the concatenation of a query-document pair is a well-established approach for document ranking (Nogueira et al., 2020). Today, modern neural methods for re-ranking are based on the encoder-only (e.g., BERT (Devlin et al., 2019)) or encoder-decoder (e.g., T5 (Raffel et al., 2020)) paradigm where query-document interactions are modeled by the encoder's attention mechanism. Unfortunately, these paradigms are computationally prohibitive given that the model has to be run on all document-query pairs during inference. To this end, it is commonplace to use less powerful but computationally lightweight dual encoder models (Nogueira et al., 2019a; Karpukhin et al., 2020; Xiong et al., 2020; Qu et al., 2021; Gao et al., 2021) for first-pass retrieval and to only run the more expensive re-ranker on a small subset of retrieved candidates. Even with this setup, cross-attention-based re-ranking can still be expensive, especially when larger pretrained Transformer models are used. As such, this paper is primarily concerned with improving inference-time re-ranking efficiency while maintaining comparable effectiveness to existing cross-attention models.

The novelty of this paper lies in a new paradigm for re-ranking that provides up to 6.8X speedup without any degradation in shallow-pool effectiveness. Concretely, we propose a new method for inference-time decomposition of encoder-decoder architectures into decoder-only language models. Given a pretrained sequence-to-sequence model, we finetune the encoder-decoder model using a document-to-query multi-task loss. At inference, we decompose the encoder-decoder architecture into a decoder-only language model (LM) that learns to interpret from a memory store of encoded document tokens representations using attention. The document-query pair score can be interpreted as the likelihood of generating the query given the encoded document term representations.

There are multiple efficiency benefits to our proposed design. First, significant inference-time cost savings are unlocked since the document term memory store can be pre-computed in advance and act as a read-only memory. Second, our redesign also exploits the fact that queries are generally much shorter than documents. During inference time, only query tokens have to be passed through the decoder stack when attending to the pre-computed document representations which allows us to also obtain an additional speed advantage over encoder-only BERT-like models. Third, computing the query likelihood is computationally simple and does not require the typical costs associated with autoregressive generation models.

The overall contributions of this work can be summarized as follows:

- We propose a new re-ranking paradigm, ED2LM (Encoder-Decoder to Language Model) for fast and efficient inference-time re-ranking. Our method is based on inference-time decomposition of an encoder-decoder model into a decoder-only language model.

- The proposed method utilizes a new fine-tuning paradigm by incorporating a new objective function that combines the generative query likelihood and the discriminative cross-entropy loss.

- Via extensive experiments, we show that the proposed method performs competitively with T5-based cross-attention re-rankers (Nogueira et al., 2020) while being up to more than 6.8X faster during inference.

2 Related Work

Neural text ranking. A number of so-called cross-attention models concatenate a query and a candidate document into a string and feed it into the model (Han et al., 2020; Nogueira et al., 2020), which allows the attention mechanism of the model to capture interactions across query and document terms. However, deploying such models to millions or billions of documents is usually intractable due to the exorbitant computational cost. To combat this cost, other studies have explored more efficient models, e.g., dual-encoder models (Karpukhin et al., 2020; Qu et al., 2021; Ren et al., 2021), BERT with late interaction (Khattab and Zaharia, 2020), or using contextual language models to improve term weighting in traditional inverted indexes (Nogueira et al., 2019a; Dai and Callan, 2020; Gao et al., 2021).

A few studies that are most closely related to this work focus on leveraging the generative nature of pretrained encoder-decoder language models. A natural practice is to directly use the likelihood of generating the query given a document to rank the documents (Zhuang and Zuccon, 2021; Zhuang et al., 2021b; Lesota et al., 2021). However, these methods mostly perform substantially worse than cross-attention ranking models. Another work (dos Santos et al., 2020) transforms the likelihood of generating the query into a discriminative loss, where an "unlikelihood" loss is introduced for negative query-document pairs. Despite relatively better performance than using vanilla maximum likelihood estimation (MLE), we found that their method still underperforms cross-attention ranking models. Our proposed method uses a combination of query generation loss and a cross-entropy loss on a specific token, which is capable of achieving comparable performance to cross-attention models.

Other work (Ju et al., 2021) uses query generation as an auxiliary task during training and shows improved performance. However, the proposed model still takes both a query and a document as input in the main ranking task and hence would be as costly as cross-attention ranking models during inference.

Efficient neural IR. Due to the excessive computational cost of inference in pretrained language models, there is a series of studies aiming to improve the efficiency.

A major trend is to distill expensive models into cheaper ones (Hinton et al., 2015; Sanh et al., 2019). Some distillation approaches have specifically focused on text ranking applications (Zhang et al., 2020; Zhuang et al., 2021a; Chen et al., 2021a; Hofstätter et al., 2021).

Another trend is to improve model efficiency by modifying the model architecture. A typical approach used by ColBERT (Khattab and Zaharia, 2020) and PreTTR (MacAvaney et al., 2020) defer query-document interactions to upper layers so that part of the model can be pre-computed. Our model can be categorized into this class of models, except that the late interaction is naturally aligned with the decomposition of encoder-decoder models. This alignment allows us to better leverage knowledge learned by the model during pretraining, and can be the reason behind our stronger performance compared to ColBERT and PreTTR.

There are a couple of other efficient model structures, such as early exiting (Soldaini and Moschitti, 2020; Xin et al., 2020), Transformer-Kernel (TK) model (Hofstätter et al., 2020), and contextualized offline relevance weighting (Chen et al., 2021b). In terms of storage cost, Cohen et al. (2021) proposed the succinct document representation which reduces the dimension of token representation to compress document representations. These techniques are orthogonal to our study and can be combined with our work to further improve the time and storage efficiency.

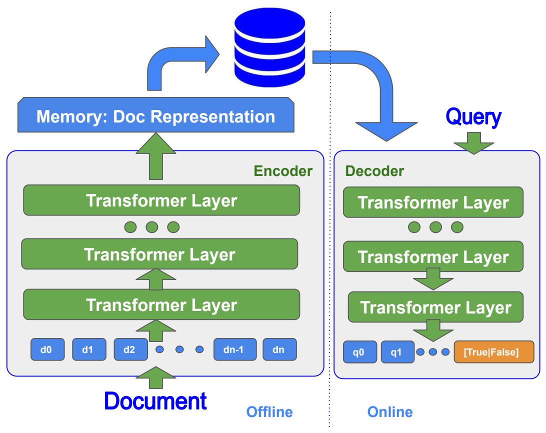

Figure 1: Overview of the proposed ED2LM.

3 The Proposed Method

This section describes the ED2LM model. See Fig. 1 for an overview of the approach.

3.1 Overview

The proposed ED2LM model is based on the T5 encoder-decoder architecture. It encodes the documents without looking at the queries and produces ranking scores by decoding the queries and attending to the document representations.

In particular, for a query-document pair, the document tokens are encoded with a stack of Transformer layers as in BERT (Devlin et al., 2019), where the tokens attend to one another before going through the position-wise feed-forward layer. The output of the encoder is in the form of dense representations for the document tokens. During decoding, the query tokens are decoded with a stack of decoder layers, where the query tokens first attend to other query tokens before going through a multi-head attention block to attend to the document tokens from the encoder.

Inspired by T5 (Nogueira et al., 2020) for ranking and the use of BART for discrimination (dos Santos et al., 2020; Lewis et al., 2020), a special true/false token is appended to the end of the query before the end of the query sequence (EOS). During training, inspired by (Ju et al., 2021), the model is trained to generate the query tokens and determine the relevance of the query-document pair. During inference, only the score for the true/false token is used for ranking.

3.2 ED2LM for Re-ranking

In this section, we describe the details of training and inference for ED2LM.

3.2.1 Fine-tuning

During fine-tuning, ED2LM involves an encoder-decoder architecture which maps $\mathbb{R}^{L_D}$ discrete symbols to $\mathbb{R}^{L_Q}$ discrete symbols. Here, L_D refers to the length of the document and L_Q refers to the query length.

Task Formulation The input to the model is a sequence of document tokens and the output of the model is a sequence of query tokens. In order to imbue our model with discriminative capabilities, we append the class token (true/false) that represents the query-document pair at the end of the query. The ranking score of a query-document pair is the normalised probability of the true token at the end of the query. Given a query q and a document d, the ground-truth correctness of d relative to q is denoted as a binary label y.

Loss function. The loss function optimized for fine-tuning has two components. The first component is the maximum likelihood estimation (MLE) loss of the individual question tokens, which is defined as:

$$Loss_{QL} = -\sum_{i \in 0 \cdots L_Q - 1} log(P(q_i | q_{:i}; d)) \quad (1)$$

Since we want the model to learn the correctness of the question using the trailing true/false tokens, we also compute the likelihood of those tokens as follows.

$$p^+ = P(\text{true,eos}|q; d)$$
$$p^- = P(\text{false,eos}|q; d)$$

The cross-entropy loss $Loss_{CE}$ can then be written as:

$$Loss_{CE} = -y log p^+ - (1-y) log p^- \quad (2)$$

The final training loss can the be written as:

$$Loss = Loss_{CE} + y Loss_{QL} \quad (3)$$

The cross-entropy loss is applied to all examples whereas the query likelihood loss only applies to the positive examples. Our fine-tuning loss is trained with teacher forcing.

Scoring. The normalised scores from the true and false tokens are combined as in (Nogueira et al., 2020).

3.3 Efficient Re-ranker

This section discusses using ED2LM for more efficient inference, by decoupling the encoder-decoder into a decoder-only language model.

3.3.1 Decomposing Encoder-Decoder to Decoder-only LM

The key idea for fast inference is to only extract the decoder from the trained Encoder-Decoder model. Recall a decoder-stack is comprised of decoder-side causal self-attention and encoder-decoder cross-attention.

$$X'_\ell = \mathrm{CausalSelfAttention}(X_\ell, X_\ell) \quad (4)$$

$$Y_\ell = \mathrm{MultiheadAttention}(M_\ell, X'_\ell) \quad (5)$$

where $X \in \mathbb{R}^{L_Q \times d_{model}}$ is the input to the decoder stack at layer ℓ. M refers to a sequence of memory tokens. In this case, we note that M here refers to computed encoder representations that pass through the encoder-stack. During fine-tuning, this encoder-stack is trained end-to-end. However, this paradigm generalizes these embeddings as *"memory"*, which can be extended to other use cases or applications. We can also interpret this memory as a form of soft prompt.

3.3.2 Reading from Memory

The decoder reads from M. In the standard setup, M are static representations that originate from the final output of the encoder in the Seq2Seq architecture and the $\mathrm{MultiheadAttention}$ is the encoder-decoder cross attention. Here, M can be compressed along the presentation dimension (d_{model}) as in (MacAvaney et al., 2020; Gao et al., 2021; Cohen et al., 2021), which is orthogonal to our studies, or along the sequence dimension (L_D), which is introduced below. We find that this generalization is a practically useful way to interpret the ED2LM architecture. We propose to explore not only standard M from encoder outputs but also compressed memory stores from Funnel Transformers (Dai et al., 2020). Herein, we employ the Funnel Transformer with b blocks in the encoder, leading to 2^b storage compression, by reducing the $\mathbb{R}^{L_D}$ for 2^b. Between each block, a mean-pooling layer is used to downsample the input sequence by two in the sequence length dimension.

4 Experiment Setup

This section describes our experimental setup.

Dataset and metrics. We employ the MS MARCO (Nguyen et al., 2016) passage re-ranking task, for which we report the official evaluation metric MRR@10 on the 6980 development queries using the binary labels from the dev dataset. We also use the 43 test queries from the TREC Deep Learning (DL) Track 2019 (Craswell et al., 2020) and the 54 test queries from 2020 (Craswell et al., 2021). The TREC data sets include graded relevance judgments. We report the official evaluation metrics NDCG@10 as well as mean average precision (MAP). When computing MAP, following the official TREC setup, we map passage judgments 2 and 3 to relevant and 0 and 1 to non-relevant. Statistical significance is reported using a paired two-tailed t-test. We use a maximum sequence length of 256 tokens for paragraphs and 32 tokens in our experiments, similar to (Hofstätter et al., 2020; Hofstätter et al., 2021).

We employ the training data from RocketQA (Qu et al., 2021), which is derived from the MS MARCO training dataset as dual-encoder models trained on it demonstrate strong performance. Specifically, we use the hard-question split ("RQA-Hard"), which only includes the hard-negative samples and positive samples from MS MARCO, and the merge split ("RQA-Merge"), which includes extra unlabeled questions from Yahoo! Answers[1], ORCAS (Fisch et al., 2019), and Natural Questions (Kwiatkowski et al., 2019) on top of "RQA-Hard". We also train all models on the original MS MARCO training dataset, where the positive and negative classes are balanced by up-sampling the positive training samples. For validation purposes, we use the 1500 dev2 validation queries with at least one relevance judgment from the TREC DL Track 2021[2]. Given our focus on shallow-pool effectiveness, the model with highest MRR@10 on the validation dataset is selected. We employ Mesh Tensorflow (Shazeer et al., 2018) for training and evaluation. The T5 models have been trained and inferred as in (Nogueira et al., 2020), and ED2LM has been primarily trained using the loss defined in Eq. 3. We train models for ablation study by using Eq. 1 and Eq. 2 separately. During training, a constant learning rate of $1e$-3 is used.

[1]http://answers.yahhoo.com

[2]https://msmarco.blob.core.windows.net/msmarcoranking/passv2_dev2_queries.tsv

Baselines. ED2LM is compared to ranking models using four variants of T5 (T5-small, T5-base, T5-large, and T5-xl), BERT-base, BERT-large, and PreTTR (MacAvaney et al., 2020). The PreTTR (MacAvaney et al., 2020) model decouples the encoding of the query and the document on top of the BERT architecture and is directly comparable to the T5-based ED2LM. We fine-tune BERT-base and BERT-large models using TF-ranking (Pasumarthi et al., 2019) and achieve similar results with the results reported in (Nogueira et al., 2020). The BERT-base is included in our result table, and the comparisons relative to BERT-large are available in the appendix (Table 4). We also re-implement the PreTTR model using TF-ranking (Pasumarthi et al., 2019). Therein, following the configurations in (MacAvaney et al., 2020), a query and a document are encoded independently in the first l-layers using the BERT-base configuration before interacting via cross-attention. The BERT-base pre-trained checkpoint is used for initialisation. We report the results by setting $l = 6$, which leads to similar FLOPs and latency as ED2LM-base (26.1T vs 20.6T). We also experiment with $l = 3$ and 9, whose results are included in the appendix (Table 4).

Variants of ED2LM. We investigate the effectiveness and inference efficiency of ED2LM based on T5-small, T5-base, T5-large, and T5-xl architectures, leading to ED2LM-small, ED2LM-base, ED2LM-large, and ED2LM-xl, respectively. We experiment with two Funnel-Transformer variants, where two six-layers funnel blocks ($b = 2$) and three eight-layers funnel blocks ($b = 3$) are used in the encoder, respectively. They are named ED2LM-F-$6L^{\times 2}$ and ED2LM-F-$8L^{\times 3}$, correspondingly. These configurations lead to a 4X (when $b = 2$) and a 8X (when $b = 3$) reduction in the sequence length. The Funnel-Transformer variants are pre-trained using the same task as in T5 on top of the C4 corpus (Raffel et al., 2020).

Initial rankings. Since we primarily focus on the re-ranking setting, we consider several retrieval models to generate initial ranking candidates. For the MS MARCO passage re-ranking task, we use BM25 (an implementation from Terrier (Macdonald et al., 2012)) to generate the top-1K passages per query. In addition, we implemented the docT5query model (Nogueira et al., 2019b,a) by training a T5 seq2seq model to generate 40 questions (i.e., expansions) per paragraph and use BM25 to retrieve top-1K passages. This serves as a high-recall initial ranking, wherein the recall@1K increases from 86.7 (MRR@10=19.3) in the base BM25 ranking to 93.76 (MRR@10=25.3) with document expansion. For the TREC DL Track, we use the official top-1k initial rankings from BM25 (Craswell et al., 2020, 2021).

Efficiency metrics. To compare inference efficiency, we report FLOPs and latency as encouraged by Dehghani et al. (2021). To compute FLOPs we make use of a public repository [3]. To compute latency, we do as follows: each model is exported in the Tensorflow Saved Model format before serving via the Tensorflow Model Server [4] on a Intel Xeon CPU desktop with 8 CPU cores, 16 CPU threads, and 132 GB RAM. We randomly select 500 queries and passages from the MS MARCO dataset. As for PreTTR (MacAvaney et al., 2020), to enable fair comparisons, we add an additional 500 queries, leading to a total of 1000 query-passages pairs, to fully utilise the shared computation of the query encoder. For each query-passage pair, we time the inference call to the model server 10 times and record the minimum. For each model, we report the 50 and 95-percentile of the 500 timing (1000 for PreTTR) as a two-number summary of latency. The time for tokenization is included for all models. For PreTTR and ED2LM, we assume the token representations of passages have already been loaded in the memory akin to (MacAvaney et al., 2020; Gao et al., 2021).

5 Results

In this section, we examine the effectiveness-efficiency trade-off of ED2LM on the passage re-ranking task. The results of T5, ED2LM, BERT, and PreTTR have been displayed in Table 1. In Table 2, we further summarise the comparisons (ED2LM vs. baseline models) from Table 1 and highlight the results that ED2LM provides a better trade-off. We also visualise the results from different models on the MS MARCO benchmark in Fig. 2 when using docT5query (Nogueira et al., 2019a) as the initial ranking.

Results for the baseline models. We achieve comparable results as previous studies on all three

[3]https://github.com/google-research/electra/blob/master/flops_computation.py
[4]https://www.tensorflow.org/tfx/tutorials/serving/rest_simple

Models	MS MARCO (MRR@10)		Trec DL Track 2019		Trec DL Track 2020		FLOPs	Latency (ms)	
	BM25+	docT5query+	nDCG@10	MAP	nDCG@10	MAP	(T)	P50	P95
ColBERT (Khattab and Zaharia, 2020)	34.9	-	-	-	-	-			
COIL (Gao et al., 2021)	34.8	-	-	-	-	-			
Baseline Models									
PreTTR (p)	36.7	37.4	70.0	39.8	71.5	45.5	26	159	189
BERT-base (b)	36.5	37.2	68.5	41.9	71.9	45.7	52	309	443
T5-small ($t5s$)	35.9	36.6	68.8	42.3	68.1	42.1	22	123	127
T5-base ($t5b$)	38.3	39.2	71.1	43.1	73.7	48.6	67	405	425
T5-large ($t5l$)	39.4	40.3	72.0	42.9	73.0	48.0	202	1111	1140
T5-xl ($t5x$)	39.6	40.6	71.8	42.2	74.6	49.2	752	2490	2515
Variants of ED2LM									
ED2LM-small	37.2 ($\uparrow_{t5s}\downarrow_{t5blx}\uparrow_{b}$)	37.9 ($\uparrow_{t5s}\downarrow_{t5blx}\uparrow_{b}$)	69.5 ($\downarrow_{t5l}$)	40.8	69.6 ($\downarrow_{t5blx}$)	43.3 ($\downarrow_{t5blx}\downarrow_{b}$)	5	60	65
ED2LM-base	38.7 ($\uparrow_{t5s}\downarrow_{t5lx}\uparrow_{b}\uparrow_{p}$)	39.6 ($\uparrow_{t5s}\downarrow_{t5lx}\uparrow_{b}\uparrow_{p}$)	70.2	42.5 ($\uparrow_{p}$)	71.5 ($\uparrow_{t5s}\downarrow_{t5x}$)	47.2 ($\uparrow_{t5s}\downarrow_{t5x}$)	21	157	185
ED2LM-large	38.0 ($\uparrow_{t5s}\downarrow_{t5lx}\uparrow_{b}\uparrow_{p}$)	39.0 ($\uparrow_{t5s}\downarrow_{t5lx}\uparrow_{b}\uparrow_{p}$)	70.3	42.3 ($\uparrow_{p}$)	72.8 ($\uparrow_{t5s}$)	47.6 ($\uparrow_{t5s}$)	73	317	336
ED2LM-xl	39.4 ($\uparrow_{t5sb}\uparrow_{b}\uparrow_{p}$)	40.4 ($\uparrow_{t5sb}\uparrow_{b}\uparrow_{p}$)	71.4	44.8 ($\uparrow_{t5sbx}\uparrow_{b}\uparrow_{p}$)	71.6 ($\uparrow_{t5s}\downarrow_{t5x}$)	48.2 ($\uparrow_{t5s}\uparrow_{b}\uparrow_{p}$)	287	811	834
ED2LM with Funnel Blocks									
ED2LM-F-6$L^{\times 2}$	36.5 ($\downarrow_{t5blx}$)	37.4 ($\uparrow_{t5s}\downarrow_{t5blx}$)	68.0 ($\downarrow_{t5blx}$)	40.5 ($\downarrow_{t5b}$)	70.4 ($\downarrow_{t5bx}$)	44.1 ($\downarrow_{t5blx}$)	9	130	151
ED2LM-F-8$L^{\times 3}$	35.4 ($\downarrow_{t5blx}\downarrow_{b}\downarrow_{p}$)	36.2 ($\downarrow_{t5blx}\downarrow_{b}\downarrow_{p}$)	69.2 ($\downarrow_{t5l}$)	40.2 ($\downarrow_{t5bl}$)	70.5 ($\downarrow_{t5bx}$)	44.7 ($\downarrow_{t5blx}$)	7	108	126

Table 1: The re-ranking performance when re-ranking top-1K paragraphs. We note down the significant difference at 0.05 level with ↑ and ↓ for the variants of ED2LM. The comparisons are relative to T5-small, T5-base, T5-large, and, T5-xl (with subscriptions $t5s$, $t5b$, $t5l$, $t5x$), BERT-base (with subscriptions b), PreTTR with six layers of decoupled encoding (with subscriptions p).

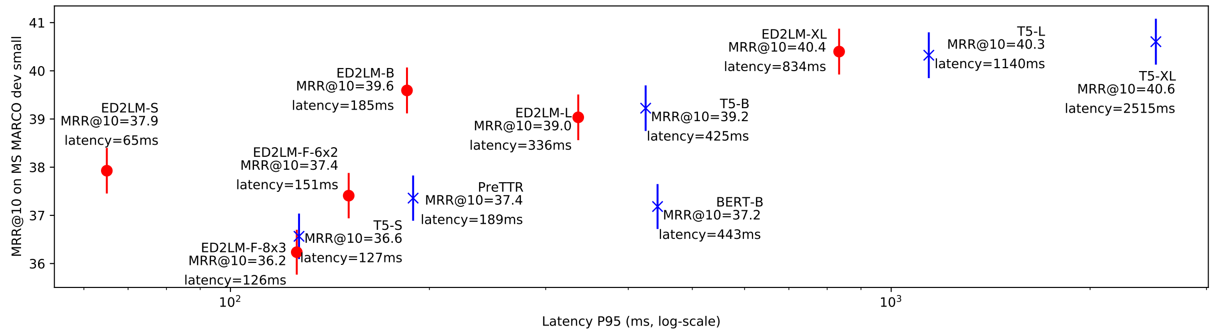

Figure 2: MRR@10 on MS MARCO dev small (6980 test queries) after re-ranking top-1K documents from docT5query (Nogueira et al., 2019a) vs. latency. The x-axis is the latency (95 percentile out of 500 calls); y-axis is the MRR@10 score. The point (ED2LM models) and the cross (baseline models) are the mean MRR@10 and the bar indicates the 95% confidence interval.

benchmarks. In particular, (Nogueira et al., 2020) reports $MRR@10 = 37.2, 38.1, 39.3$, and 39.8 when using BERT-large, T5-base, T5-large, and T5-xl to re-rank top-1K paragraphs from BM25 on MS MARCO passage re-ranking benchmark. Besides, we also include the re-ranking results from COIL (Gao et al., 2021) and ColBERT (Khattab and Zaharia, 2020). For the TREC DL Track, we select the submitted runs that are most comparable to ours, namely, the top re-ranking run (Yan et al., 2019) in 2019 ($nDCG@10 = 72.5$ and $MAP = 45.3$) and the 4th best re-ranking run (Cao et al., 2020)[5] for 2020 ($nDCG@10 = 73.7$ and $MAP = 48.8$). It is worth mentioning that the former run employs customised pre-training methods for BERT-large, whereas the latter uses ensemble models, thus achieving slightly higher results than our T5-variants.

[5]The 1st-3rd best runs (Qiao et al., 2021) in 2020 used TREC DL 2019 data for fine-tuning.

Effectiveness-efficiency trade-off. ED2LM decouples the encoding of the document and query, thereby allowing for caching the document representation offline. After pre-computing the document presentation as in PreTTR (MacAvaney et al., 2020), ED2LM achieves a highly favorable trade-off. From Table 1 and 2, we make the following observations. (1) ED2LM-small and ED2LM-base perform at least as good as T5-small and T5-base, respectively, while providing more than a 2X speed up. For ED2LM-base, its effectiveness is not significantly different from T5-large on both TREC DL Tracks and under-performs by 0.7 (38.7 vs 39.4) on MS MARCO, while providing a 6.2X speed up. When comparing with BERT-base and PreTTR,

ED2LM→	Small	Base	Large	xl	F-6$L^{\times 2}$	F-8$L^{\times 2}$
T5-small	F:4.4x/L:2.0x r:↑/n:~/m:~	-	-	-	F:2.4x/L:0.8x r:~/n:~/m:~	**F:3.1x/L:1.0x r:~/n:~/m:~**
PreTTR	F:5.2x/L:2.9x r:~/n:~/m:~	F:1.2x/L:1.0x r:↑/n:~/m:↑	-	-	F:2.9x/L:1.3x r:~/n:~/m:~	F:3.7x/L:1.5x r:↓/n:~/m:~
BERT-base	**F:10.4x/L:6.8x r:↑/n:~/m:↓**	F:2.5x/L:2.4x r:↑/n:~/m:~	-	-	**F:5.8x/L:2.9x r:~/n:~/m:~**	F:7.4x/L:3.5x r:↓/n:~/m:~
T5-base	-	**F:3.2x/L:2.3x r:~/n:~/m:~**	-	-	-	-
T5-large	-	F:9.6x/L:6.2x r:↓/n:~/m:~	F:2.8x/L:3.4x r:↓/n:~/m:~	-	-	-
T5-xl	-	-	F:10.3x/L:7.5x r:↓/n:~/m:~	F:2.6x/L:3.0x r:~/n:↓/m:~	-	-

Table 2: The comparison of the effectiveness-efficiency trade-off for ED2LM derived from Table 1. Each row includes one baseline model, and individual columns are one of the ED2LM variants. In each comparison (cell), the upper part is the efficiency comparison, where F indicates FLOPs and L is the latency (P95). In the lower part, the comparisons for the effectiveness are summarised. ↑, ↓, and, ~ denote the significant better, worse, and, no significant difference (at level 0.05) when comparing ED2LM models with the baseline. Herein, r indicates MRR@10 on MS Marco dev small dataset (re-ranking top-1k from BM25); n and m denote nDCG@10 and MAP, respectively, on TREC DL Track. We list comparisons that ED2LM could provide better effectiveness (MRR@10 or nDCG@10) or smaller latency. The full results can be found in Table 1 and in the Appendix (Table 4).

both ED2LM-small and ED2LM-base perform at least as good (for MRR@10 and nDCG@10) and are up to 6.8X faster. (2) ED2LM-large performs on par with T5-large on the TREC DL Tracks, but under performs on MS MARCO by 1.4; whereas ED2LM-xl achieves similar MRR@10 on MS MARCO (39.4 vs 39.6), but performs worse in terms of nDCG@10 on TREC DL Track 2020. Furthermore, in Fig. 2 (MRR@10 on MS MARCO vs the latency (P95) by re-ranking the top-1K from docT5query) the leftmost ED2LM-small achieves better effectiveness than T5-small, PreTTR, and BERT-base. Likewise, ED2LM-base achieves similar latency as PreTTR and is 2.3X more efficient than BERT-base but achieves higher MRR@10. In the meantime, though more efficient, ED2LM-xl and ED2LM-large perform close to their counterparts, once again confirming the observations. We argue that, on the one hand, co-training of query likelihood and the discriminative cross-entropy leads to better ranking quality, which is especially true for the smaller variants (small and base); On the other hand, not attending to the query during document encoding leads to performance decreases, which dominates the outcomes in larger model variants (like large and xl).

ED2LM-F: Storage compression with Funnel Transformer. The results for the two variants of ED2LM with Funnel blocks are summarised in the bottom block of Table 1 and the rightmost columns in Table 2. In terms of storage, ED2LM-F-6$L^{\times 2}$ provides 4X compression and ED2LM-F-8$L^{\times 3}$ provides 8X compression by reducing the sequence length in the encoder. It can be seen that, ED2LM-F-6$L^{\times 2}$ outperforms T5-small and performs as well as BERT-base and PreTTR. Furthermore, while ED2LM-F-8$L^{\times 3}$ provides 8X compression, the effectiveness drops below that of T5-small and BERT-base on the MS MARCO benchmark. However, it achieves on-par results relative to T5-small and BERT-base on the TREC DL Track in terms of both nDCG@10 and MAP. As for efficiency, ED2LM-F-8$L^{\times 3}$ is similar to T5-small and PreTTR, but is 3.5X faster than BERT-base.

5.1 Analysis

The use of RocketQA-Merge dataset for training. In our experiments, we find that the ranking quality of the proposed ED2LM, as well as PreTTR model, benefit considerably from RocketQA-Merge. We demonstrate the training performance (upper part) in Table 3 on RocketQA and the MS MARCO training dataset. It can be seen that T5 achieves similar performance on both training data sets. In the meantime, ED2LM achieves MRR@10=37.5 when trained on the MS MARCO training dataset, and can

Models	MS Marco		
	Training Data	Loss	MRR@10
PreTTR	MS Marco	-	35.2
T5-base	MS Marco	-	38.4
T5-base	RQA-Hard	-	38.0
ED2LM-base	MS Marco	-	37.5
ED2LM-base	RQA-Hard	-	37.3
ED2LM-base	MS Marco	LUL (dos Santos et al., 2020)	31.2
ED2LM-base	RQA-Merge	LUL (dos Santos et al., 2020)	33.6
ED2LM-base	RQA-Merge	MLE (Eq. 1)	30.2
ED2LM-base	RQA-Merge	CE (Eq. 2)	38.2

Table 3: Ablation study. In the upper half, the uses of alternative training data are explored. In the lower half, different loss functions are used to train ED2LM, including the LUL loss from (dos Santos et al., 2020), negative log-likelihood loss on questions as in (Nogueira et al., 2019a), and the cross-entropy loss on true/false token as in (Nogueira et al., 2020).

achieve 38.7 when trained on the "RQA-Merge" dataset. This is also true for PreTTR, which sees an MRR@10 increase from 35.2 to 36.7. We conjecture that the decoupled encoding of query and documents, as in ED2LM and PreTTR, requires more queries for training whereas models that use full cross-attention benefit less from the extra training data. The training performance of ED2LM-base on RocketQA-Hard in Table 3 provides evidences for this, where ED2LM-base achieves an even lower MRR@10. RocketQA-Hard is a subset of RocketQA-Merge and includes hard negative samples but without the extra queries. Therefore, we conclude that *more unique questions for training is one of the ED2LM's key ingredients.*

Alternative loss functions for training. In (dos Santos et al., 2020), the unlikelihood loss (referred as LUL) was used to train a BART (Lewis et al., 2020) model for question answering. In this section, we train ED2LM using the LUL loss from (dos Santos et al., 2020) on both the MS MARCO and RQA-Merge training sets. We also use the negative log-likelihood loss in Eq. 1 (as in docT5query (Nogueira et al., 2019a)) and the cross-entropy loss in Eq. 2 (as in (Nogueira et al., 2020)) to train ED2LM separately. From Table 3 (lower part), LUL leads to significantly worse MRR@10 than using the loss in Eq. 3 (33.6 vs 38.7), but outperforms the use of negative log-likelihood loss from Eq. 1 as in (Zhuang et al., 2021b). When only using the cross-entropy loss of the true/false token (Eq. 2), effectiveness is slightly worse than when using the loss in combination with query likelihood (38.2 vs 38.7), mirroring the findings from (Ju et al., 2021). Therefore, we conclude that *the use of both true/false tokens and query likelihood for training (as in Eq. 3) is another key ingredient for ED2LM.*

Manual inspection of the generated questions. We further investigate the reasons why ED2LM can significantly outperform deep query likelihood (MRR@10=38.7 vs 30.2 from Table 3) by a big margin. We compare the questions generated by ED2LM and T5 trained with query likelihood as in Eq. 1. We sample 66 documents from the MS MARCO passage corpus with at least one correct query in the MS MARCO development dataset, and collect 10 unique generated queries from both ED2LM and T5, ending up with 660 query-documents pairs for annotation. These pairs are labeled by eight annotators with a single binary question: "Is the generated query (question) answered by the given document (passage)?". We avoid potential bias during annotation by not informing the annotators which system generated which questions. According to the annotated data, **70.6%** of the queries generated by ED2LM are answerable by the source document, while **52.1%** of the queries generated by T5 are answerable. We conjecture that *the use of Eq. 3 for training makes the query generator stick to the document better, leading to fewer hallucinations, thus producing better ranking when the decoder is used as a ranker.* Configuration details and more analyses for the question generation can be found in the Appendix (Section A.2).

6 Conclusion

In this work, we propose a novel re-ranking model named ED2LM, that works by finetuning an encoder-decoder model. ED2LM encodes documents and decodes the query using a trailing binary class token appended to the query for ranking. By training on a dataset with more unique questions (namely, "RocketQA-Merge" (Qu et al., 2021)) and optimizing both query likelihood and a discriminative loss over the true/false token, ED2LM achieves competitive results compared to corresponding T5 models. When used as a decoder-only language model during inference, ED2LM provides up to 6.8X speedup without sacrificing effectiveness. It was also shown that Funnel-Transformer (Dai et al., 2020), when used in conjunction with ED2LM, can compress the storage of the pre-computed memory, making ED2LM a good modeling choice when it comes to the efficiency and effectiveness tradeoff.

References

Liyu Cao, Yixuan Qiao, Hao Chen, Peng Gao, Yuan Ni, and Guotong Xie. 2020. A multiple models ensembling method in trec deep learning. In *TREC*.

Xuanang Chen, Ben He, Kai Hui, Le Sun, and Yingfei Sun. 2021a. Simplified TinyBERT: Knowledge distillation for document retrieval. In *Advances in Information Retrieval - Proceedings of the 43rd European Conference on IR Research, Part II*, volume 12657 of *Lecture Notes in Computer Science*, pages 241–248. Springer.

Xuanang Chen, Ben He, Kai Hui, Yiran Wang, Le Sun, and Yingfei Sun. 2021b. Contextualized offline relevance weighting for efficient and effective neural retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1617–1621.

Nachshon Cohen, Amit Portnoy, Besnik Fetahu, and Amir Ingber. 2021. SDR: Efficient neural re-ranking using succinct document representation. *arXiv preprint arXiv:2110.02065*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the trec 2020 deep learning track.

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

Zhuyun Dai and Jamie Callan. 2020. Context-aware term weighting for first stage passage retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1533–1536.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. In *NeurIPS*.

Mostafa Dehghani, Anurag Arnab, Lucas Beyer, Ashish Vaswani, and Yi Tay. 2021. The efficiency misnomer. *arXiv preprint arXiv:2110.12894*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*.

Cicero dos Santos, Xiaofei Ma, Ramesh Nallapati, Zhiheng Huang, and Bing Xiang. 2020. Beyond [cls] through ranking by generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1722–1727.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. Mrqa 2019 shared task: Evaluating generalization in reading comprehension. *arXiv preprint arXiv:1910.09753*.

Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. Coil: Revisit exact lexical match in information retrieval with contextualized inverted list. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3030–3042.

Shuguang Han, Xuanhui Wang, Mike Bendersky, and Marc Najork. 2020. Learning-to-rank with BERT in tf-ranking. *CoRR*, abs/2004.08476.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Sebastian Hofstätter, Markus Zlabinger, and Allan Hanbury. 2020. Interpretable & time-budget-constrained contextualization for re-ranking. In *ECAI 2020*, pages 513–520. IOS Press.

Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2021. Improving efficient neural ranking models with cross-architecture knowledge distillation.

Jia-Huei Ju, Jheng-Hong Yang, and Chuan-Ju Wang. 2021. Text-to-text multi-view learning for passage re-ranking. *arXiv preprint arXiv:2104.14133*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

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*, pages 39–48.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Oleg Lesota, Navid Rekabsaz, Daniel Cohen, Klaus Antonius Grasserbauer, Carsten Eickhoff, and Markus Schedl. 2021. A modern perspective on query likelihood with deep generative retrieval models. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 185–195.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Jing Lu, Gustavo Hernandez Abrego, Ji Ma, Jianmo Ni, and Yinfei Yang. 2021. Multi-stage training with improved negative contrast for neural passage retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: Findings*.

Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Efficient document re-ranking for transformers by precomputing term representations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 49–58.

Craig Macdonald, Richard McCreadie, Rodrygo L. T. Santos, and Iadh Ounis. 2012. From puppy to maturity: Experiences in developing terrier. In *Proceedings of the SIGIR 2012 Workshop on Open Source Information Retrieval*, pages 60–63. University of Otago, Dunedin, New Zealand.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. In *CoCo@ NIPS*.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 708–718.

Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019a. From doc2query to docTTTTTquery. *Online preprint*.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019b. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*.

Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. 2019. Tf-ranking: Scalable tensorflow library for learning-to-rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2970–2978.

Yixuan Qiao, Hao Chen, Liyu Cao, Liping Chen, Pengyong Li, Jun Wang, Peng Gao, Yuan Ni, and Guotong Xie. 2021. Pash at trec 2020 deep learning track: Dense matching for nested ranking.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5835–5847.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. RocketQAv2: A joint training method for dense passage retrieval and passage re-ranking. *arXiv preprint arXiv:2110.07367*.

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

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*.

Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, Ryan Sepassi, and Blake Hechtman. 2018. Mesh-TensorFlow: Deep learning for supercomputers. In *Neural Information Processing Systems*.

Luca Soldaini and Alessandro Moschitti. 2020. The cascade transformer: an application for efficient answer sentence selection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5697–5708.

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*, pages 83–88.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

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*.

Wangshu Zhang, Junhong Liu, Zujie Wen, Yafang Wang, and Gerard de Melo. 2020. Query distillation: Bert-based distillation for ensemble ranking. In *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*, pages 33–43.

Honglei Zhuang, Zhen Qin, Shuguang Han, Xuanhui Wang, Mike Bendersky, and Marc Najork. 2021a. Ensemble distillation for bert-based ranking models. In *Proceedings of the 2021 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '21)*.

Shengyao Zhuang, Hang Li, and Guido Zuccon. 2021b. Deep query likelihood model for information retrieval. In *The 43rd European Conference On Information Retrieval (ECIR)*.

Shengyao Zhuang and Guido Zuccon. 2021. TILDE: Term independent likelihood moDEl for passage reranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, pages 1483–1492, New York, NY, USA. Association for Computing Machinery.

A Appendix

A.1 Full Results for Re-ranking

The full results of our re-ranking experiments using both MS MARCO and RocketQA-Merge dataset for training can be found in Table 4 with the test of statistical significance at a significance level of 0.05.

A.2 Analysis of Question Generation

For the generation task, we train the generation models, namely, ED2LM-base and T5-base, on MS MARCO training dataset, following the same setting as in (Lu et al., 2021), to enable fair comparisons. Both models employ the top-k decoding with $k = 10$. During the decoding, we employ top-k random sampling decoding and set $k = 10$, where the top-k tokens with highest probability according to the decoder are sampled. To compare the quality of the generations from ED2LM and T5, we conduct manual annotations (as described in Section 5.1), calculate token overlaps, and employ the generated questions to train a dual-encoder based passage retrieval model following the configurations from (Lu et al., 2021).

Question vs. paragraph overlap. Beside the manual annotation in the end of Section 5, we further measured the overlap between generated questions and their respective source passages using a set of 3k generated questions from each system. Intuitively, question generators that hallucinate less are more likely to stick to the text from the source paragraph. The overlap is computed as the macro-average of the question-paragraph word-level overlap, and is normalised using the length of the question. While T5-base has an overlap rate of **55.62%** (i.e., 55.62% of question tokens also appear in the source paragraph), ED2LM-base has an overlap rate of **62.14%**, which is more than 6% higher than T5 model. This result is further evidence that ED2LM sticks to the paragraph text more frequently. Although this can be seen as a problem if one wants a more diverse set of questions, it avoids hallucinations and allows for more accurate questions, as demonstrated in the manual inspection. In Table 5, we present some examples of typical questions generated by both T5 are ED2LM and the respective source paragraph. Although T5 questions are somewhat related to the paragraph, the paragraph is not a good answer for them. Notice that in the first question T5 hallucinates the word *English*, which completely compromises the question quality.

Synthetic Training Data for Retrieval Finally, we demonstrate the advantages of the generated questions from ED2LM by using them to train a dual-encoder based passage retrieval model, following the configurations in (Lu et al., 2021). Specifically, we train a $BERT_{large}$ dual encoder model using the synthetic question-passage pairs generated by ED2LM-base and T5-base respectively and report the results on MS MARCO dev set. For each passage, we generate three synthetic questions. We also extract hard negatives by randomly sample passages from the same document. During training, we use both in-batch negatives and hard negatives. During inference, we retrieve top-1K passages for each question from the passage collection containing about 8.8 million passages and report MRR@10. The model using generated data from ED2LM achieves **MRR@10=30.4**, whereas the model using generated data from T5 gets **MRR@10=26.5**. We argue that the boost is due to that the synthetic training data from ED2LM is with less generation hallucination (18% according to the manual annotation), thus including few training noise.

A.3 Configuration Details for Latency Computation

To compute latency, individual models are exported in the Tensorflow Saved Model format before being served via the Tensorflow Model Server [6] on a Intel Xeon CPU desktop with 8 CPU cores, 16 CPU threads, and 132 GB RAM. We randomly select 500 queries and passages from the MS Marco dataset. We observe the character length distribution for queries and passages from this sample. For queries, the [25, 50, 75, 90, 95, 99]-percentiles are [24.75, 31.5, 39.25, 49.0, 56.0, 80.04] characters, whereas the same percentiles for documents are [278.0, 318.0, 447.75, 572.1, 628.15, 838.05] characters. As for PreTTR (MacAvaney et al., 2020), to enable fair comparisons, we add an additional 500 passages per query, leading to a total of 1000 query-passages pairs, to fully utilise the shared computation of the query encoder. For each of these document/query pairs, we time the inference call to the model server 10 times and record the minimum time t across the 10 runs. Let $L(\ell_e, \ell_d, s)$

[6] https://www.tensorflow.org/tfx/tutorials/serving/rest_simple

Models	MS MARCO (MRR@10)		Trec DL Track 2019		Trec DL Track 2020		FLOPs(T)	Latency (ms)	
	BM25+	docT5query+	nDCG@10	MAP	nDCG@10	MAP		P50	P95
ColBERT (Khattab and Zaharia, 2020)	34.9	-	-	-	-	-			
COIL (Gao et al., 2021)	34.8	-	-	-	-	-			
(Nogueira et al., 2020)									
BERT-large	37.2	-	-	-	-	-	-	-	-
T5-base	38.1	-	-	-	-	-	-	-	-
T5-large	39.3	-	-	-	-	-	-	-	-
T5-xl	39.8	-	-	-	-	-	-	-	-
2019 Top Run (Yan et al., 2019)	-	-	72.5	45.3	-	-	-		
2020 Top Run (Cao et al., 2020)	-	-	-	-	73.7	48.8	-	-	-
Training Data: MS MARCO (Nguyen et al., 2016) balanced triplet training data w/ 503k unique questions.									
BERT-base	36.2	36.8	68.8	41.7	69.5	45.9	52	309	443
BERT-large	37.1	37.8	71.5	45.0	71.0	48.2	114	886	929
PreTTR (l=3)	35.8	36.0	69.8	43.2	68.4	45.7	39	-	-
PreTTR (l=6)	35.2	35.6	70.1	43.2	68.6	44.9	26	159	189
PreTTR (l=9)	34.8	35.2	68.5	42.4	66.6	43.9	13	-	-
T5-small	36.3	36.5	65.6	45.0	68.7	44.6	22	123	127
T5-base	38.4	39.0	71.0	45.3	72.4	48.8	67	405	425
T5-large	39.7	40.4	72.2	46.6	73.5	50.6	202	1111	1140
T5-xl	40.3	41.2	71.8	45.8	74.6	50.6	752	2490	2515
ED2LM-small	35.5 ($\downarrow_{t5sblx}\downarrow_{bl}\uparrow_{p9}$)	35.9 ($\downarrow_{t5sblx}\downarrow_{bbl}\uparrow_{p9}$)	68.8 ($\uparrow_{t5s}\downarrow_{t5lx}\downarrow_{bl}$)	42.3 ($\downarrow_{t5sblx}\downarrow_{bl}$)	68.4 ($\downarrow_{t5blx}$)	44.8 ($\downarrow_{t5blx}\downarrow_{bl}$)	5	60	65
ED2LM-base	37.5 ($\uparrow_{t5s}\downarrow_{t5blx}\uparrow_{bb}\uparrow_{p369}$)	38.0 ($\uparrow_{t5s}\downarrow_{t5blx}\uparrow_{bb}\uparrow_{p369}$)	71.0	43.5 ($\downarrow_{t5blx}$)	68.9 ($\downarrow_{t5blx}$)	46.2 ($\downarrow_{t5blx}$)	21	157	185
ED2LM-large	37.5 ($\uparrow_{t5s}\downarrow_{t5blx}\uparrow_{bb}\uparrow_{p369}$)	37.9 ($\uparrow_{t5s}\downarrow_{t5blx}\uparrow_{bb}\uparrow_{p369}$)	70.1	44.6 ($\downarrow_{t5l}\uparrow_{bb}\uparrow_{p9}$)	71.5 ($\uparrow_{p9}$)	50.3 ($\uparrow_{t5s}\uparrow_{bb}\uparrow_{p369}$)	73	317	336
ED2LM-xl	37.8 ($\uparrow_{t5s}\downarrow_{t5lx}\uparrow_{bb}\uparrow_{p369}$)	38.5 ($\uparrow_{t5s}\downarrow_{t5lx}\uparrow_{bbl}\uparrow_{p369}$)	70.8	42.6 ($\downarrow_{t5blx}\downarrow_{bl}$)	69.6 ($\downarrow_{t5lx}\downarrow_{bl}$)	47.1 ($\downarrow_{t5lx}\uparrow_{p9}$)	287	811	834
ED2LM-F-6$L^{\times 2}$	34.2 ($\downarrow_{t5sblx}\downarrow_{bbl}\downarrow_{p36}$)	34.5 ($\downarrow_{t5sblx}\downarrow_{bbl}\downarrow_{p36}$)	65.6 ($\downarrow_{t5blx}\downarrow_{bl}$)	41.6 ($\downarrow_{t5sblx}\downarrow_{bl}$)	67.8 ($\downarrow_{t5blx}$)	45.1 ($\downarrow_{t5blx}\downarrow_{bl}$)	9	130	151
ED2LM-F-8$L^{\times 3}$	33.5 ($\downarrow_{t5sblx}\downarrow_{bbl}\downarrow_{p369}$)	33.7 ($\downarrow_{t5sblx}\downarrow_{bbl}\downarrow_{p369}$)	65.7 ($\downarrow_{t5blx}\downarrow_{bl}\downarrow_{p36}$)	40.5 ($\downarrow_{t5sblx}\downarrow_{bl}\downarrow_{p36}$)	66.9 ($\downarrow_{t5blx}\downarrow_{bl}$)	43.9 ($\downarrow_{t5blx}\downarrow_{bl}$)	7	108	126
Training Data: RocketQA-Merge (Qu et al., 2021) w/ 1.4M unique questions.									
BERT-base	36.5	37.2	68.5	41.9	71.9	45.7	52	309	443
BERT-large	36.7	37.3	69.3	39.6	70.9	45.6	114	886	929
PreTTR (l=3)	37.5	38.2	68.7	40.5	71.4	45.8	39	-	-
PreTTR (l=6)	36.7	37.4	70.0	39.8	71.5	45.5	26	159	189
PreTTR (l=9)	36.6	37.3	71.2	41.4	70.4	44.0	13	-	-
T5-small	35.9	36.6	68.8	42.3	68.1	42.1	22	123	127
T5-base	38.3	39.2	71.1	43.1	73.7	48.6	67	405	425
T5-large	39.4	40.3	72.0	42.9	73.0	48.0	202	1111	1140
T5-xl	39.6	40.6	71.8	42.2	74.6	49.2	752	2490	2515
ED2LM-small	37.2 ($\uparrow_{t5s}\downarrow_{t5blx}\uparrow_{bb}\uparrow_{p9}$)	37.9 ($\uparrow_{t5s}\downarrow_{t5blx}\uparrow_{bb}$)	69.5 ($\downarrow_{t5l}$)	40.8	69.6 ($\downarrow_{t5blx}$)	43.3 ($\downarrow_{t5blx}\downarrow_{bb}\downarrow_{p3}$)	5	60	65
ED2LM-base	38.7 ($\uparrow_{t5s}\downarrow_{t5lx}\uparrow_{bbl}\uparrow_{p369}$)	39.6 ($\uparrow_{t5s}\downarrow_{t5lx}\uparrow_{bbl}\uparrow_{p369}$)	70.2	42.5 ($\uparrow_{bl}\uparrow_{p6}$)	71.5 ($\uparrow_{t5s}\downarrow_{t5x}$)	47.2 ($\uparrow_{t5s}\downarrow_{t5x}$)	21	157	185
ED2LM-large	38.0 ($\uparrow_{t5s}\downarrow_{t5lx}\uparrow_{bbl}\uparrow_{p69}$)	39.0 ($\uparrow_{t5s}\downarrow_{t5lx}\uparrow_{bbl}\uparrow_{p369}$)	70.3	42.3 ($\uparrow_{bl}\uparrow_{p6}$)	72.8 ($\uparrow_{t5s}$)	47.6 ($\uparrow_{t5s}\uparrow_{p9}$)	73	317	336
ED2LM-xl	39.4 ($\uparrow_{t5sb}\uparrow_{bbl}\uparrow_{p369}$)	40.4 ($\uparrow_{t5sb}\uparrow_{bbl}\uparrow_{p369}$)	71.4	44.8 ($\uparrow_{t5sbx}\uparrow_{bbl}\uparrow_{p369}$)	71.6 ($\uparrow_{t5s}\downarrow_{t5x}$)	48.2 ($\uparrow_{t5s}\uparrow_{bbl}\uparrow_{p369}$)	287	811	834
ED2LM-F-6$L^{\times 2}$	36.5 ($\downarrow_{t5blx}\downarrow_{p3}$)	37.4 ($\uparrow_{t5s}\downarrow_{t5blx}\downarrow_{p3}$)	68.0 ($\downarrow_{t5blx}\downarrow_{p9}$)	40.5 ($\downarrow_{t5b}$)	70.4 ($\downarrow_{t5bx}$)	44.1 ($\downarrow_{t5blx}$)	9	130	151
ED2LM-F-8$L^{\times 3}$	35.4 ($\downarrow_{t5blx}\downarrow_{bbl}\downarrow_{p369}$)	36.2 ($\downarrow_{t5blx}\downarrow_{bbl}\downarrow_{p369}$)	69.2 ($\downarrow_{t5l}$)	40.2 ($\downarrow_{t5bl}$)	70.5 ($\downarrow_{t5bx}$)	44.7 ($\downarrow_{t5blx}$)	7	108	126

Table 4: The re-ranking performance when re-ranking top-1000 documents from BM25 (Robertson and Zaragoza, 2009) and docT5query (Nogueira et al., 2019a). The official initial rankings from Trec DL Track 2019 and 2020 are used, whereas the initial ranking in MS MARCO is created using Terrier (Macdonald et al., 2012). We also note down the significant difference at a significance level of 0.05 with $\uparrow$ and $\downarrow$. The comparisons are relative to T5-small, base, large, and, xl (with subscriptions $t5s$, b, l, x), BERT-base and large (with subscriptions bb and bl), PreTTR with $l = 3$, 6, 9 layers of decoupled encoding (with subscriptions $p3$, 6, 9).

Paragraph		
An experience modifier is an adjustment factor assigned to an Employer's FEIN by the rating bureau (NCCI or State Bureau). The factor compares your loss data to other employers with the same class codes, and is expressed as a credit or debit on your policy.		
Model	**Question**	**Answerable ?**
T5	is a modifier factor English	No
T5	what is experience modifier rating	No
ED2LM	what is an experience modifier in an insurance policy	Yes
ED2LM	experience modifier definition	Yes

Table 5: Example generations from ED2LM-base and T5-base.

represent the latency of a model with ℓ_e encoder and ℓ_d decoder layers, for any particular input with max sequence length s. For BERT-based models (which have no decoder layers) we set $\ell_d = 0$. For PreTTR, we compute an *amortized* latency. Concretely, letting $N = 1000$ be the number of samples, the amortized latency is computed as:

$$\frac{NL(6,0,256+32,0) + L(6,32,0) + NL(6,0,256,0)}{N}.$$

For encoder-decoder models we estimate $L(0, \ell_d, s)$ as

$$\frac{L(1,\ell_d,s) - [L(2,\ell_d,s) - L(1,\ell_d,s)]}{2}$$

Here, we estimate $L(6, 0, s)$ as $L(12, 0, s)/2$, using Bert Base for the latter. For each model, we report the 50 and 95-percentile of L the inputs as a two-number summary of latency. For BERT, we use the official cased English Base and Large model checkpoints. The time for tokenization is included for all models, where the official vocabularies (SentencePiece for T5-based models and WordPiece for BERT-based ones) are used. We use a maximum sequence lengths of 256 tokens for paragraphs and 32 tokens for queries.