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

Kai Hui∗ Honglei Zhuang Tao Chen Zhen Qin Jing Lu Dara Bahri Ji Ma Jai Gupta Cicero Nogueira dos Santos Yi Tay Donald Metzler

kaihuibj@google.com
Google Research

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 acts 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.** Traditional ranking systems focus on numeric input features (Qin et al., 2021; Yan et al., 2021). Recently, text ranking is popular given the prevalence of large pretrained language models. 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; Chen et al., 2022), 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.

(Ju et al., 2021) proposes 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. Finally, the recent differentiable search index (Tay et al., 2022) proposes end-to-end ranking via text generation using an encoder-decoder T5 model.

**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., 2020a).

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., 2020b), and contextualized
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:

$$
\text{Loss}_{QL} = - \sum_{i=0}^{L_Q-1} \log(P(q_i|q; d))
$$

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 $\text{Loss}_{CE}$ can then be written as:

$$
\text{Loss}_{CE} = -y \log p^+ - (1 - y) \log p^-
$$

The final training loss can then be written as:

$$
\text{Loss} = \text{Loss}_{CE} + y \text{Loss}_{QL}
$$
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.

\[ X'_{\ell} = \text{CausalSelfAttention}(X_{\ell}, X_{\ell}) \]
\[ Y_{\ell} = \text{MultiheadAttention}(M_{\ell}, X'_{\ell}) \]

where \( X \in \mathbb{R}^{L_Q \times d_{\text{model}}} \) is the input to the decoder stack at layer \( \ell \). \( M \) 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 MultiheadAttention is the encoder-decoder cross-attention. Here, \( M \) can be compressed along the presentation dimension \( (d_{\text{model}}) \) 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 for queries in our experiments, similar to (Hofstätter et al., 2020b,a).

We employ the training data from Rock- etQA (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”. 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

\(^1\)http://answers.yahoo.com
\(^2\)https://msmarco.blob.core.windows.net/msmarcoranking/passv2_dev2_queries.tsv
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.

**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 models using TF-ranking (Pasumarthi et al., 2019) and achieve similar results with the results reported in (Nogueira et al., 2020). We also re-implement the PreTTR model using TF-ranking. 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).

**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×$^2$ and ED2LM-F-8L×$^3$, 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. (2022). To compute FLOPs we make use of a public repository. To compute latency, we do as follows: each model is exported in the Tensorflow Saved Model format before serving via the Tensorflow Model Server 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.

5.1 Trade-off in Re-ranking

**Results for the baseline models.** We achieve comparable results as previous studies on all three benchmarks. In particular, (Nogueira et al., 2019b,a)
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 15s, 15b, 15l, 15x), 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.](image)

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 list the re-ranking results on MS MARCO from COIL (Gao et al., 2021) (MRR@10=34.8) and ColBERT (Khattab and Za-}

<table>
<thead>
<tr>
<th>Models</th>
<th>MS MARCO (MRR@10)</th>
<th>Trec DL Track 2019</th>
<th>Trec DL Track 2020</th>
<th>FLOPs (T)</th>
<th>Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BM25+</td>
<td>docT5query+</td>
<td>nDCG@10</td>
<td>MAP</td>
<td>nDCG@10</td>
</tr>
<tr>
<td>PreTTR (p)</td>
<td>36.7</td>
<td>37.4</td>
<td>70.0</td>
<td>39.8</td>
<td>71.5</td>
</tr>
<tr>
<td>BERT-base (b)</td>
<td>36.5</td>
<td>37.2</td>
<td>68.5</td>
<td>41.9</td>
<td>71.9</td>
</tr>
<tr>
<td>T5-small (15s)</td>
<td>35.9</td>
<td>36.6</td>
<td>68.8</td>
<td>42.3</td>
<td>68.1</td>
</tr>
<tr>
<td>T5-base (15b)</td>
<td>38.3</td>
<td>39.2</td>
<td>71.1</td>
<td>43.1</td>
<td>73.7</td>
</tr>
<tr>
<td>T5-large (15l)</td>
<td>39.4</td>
<td>40.3</td>
<td>72.0</td>
<td>42.9</td>
<td>73.0</td>
</tr>
<tr>
<td>T5-xl (15x)</td>
<td>39.6</td>
<td>40.6</td>
<td>71.8</td>
<td>42.2</td>
<td>74.6</td>
</tr>
</tbody>
</table>

**Variants of ED2LM**

<table>
<thead>
<tr>
<th>Models</th>
<th>FLOPs Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED2LM-base</td>
<td>37.2 (15s, 15l)</td>
</tr>
<tr>
<td>ED2LM-large</td>
<td>38.0 (15s, 15l)</td>
</tr>
<tr>
<td>ED2LM-xl</td>
<td>39.4 (15s, 15l)</td>
</tr>
</tbody>
</table>

**ED2LM with Funnel Blocks**

<table>
<thead>
<tr>
<th>Models</th>
<th>MS MARCO (MRR@10)</th>
<th>Trec DL Track 2019</th>
<th>Trec DL Track 2020</th>
<th>FLOPs (T)</th>
<th>Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BM25+</td>
<td>docT5query+</td>
<td>nDCG@10</td>
<td>MAP</td>
<td>nDCG@10</td>
</tr>
<tr>
<td>PreTTR (p)</td>
<td>36.7</td>
<td>37.4</td>
<td>70.0</td>
<td>39.8</td>
<td>71.5</td>
</tr>
<tr>
<td>BERT-base (b)</td>
<td>36.5</td>
<td>37.2</td>
<td>68.5</td>
<td>41.9</td>
<td>71.9</td>
</tr>
<tr>
<td>T5-small (15s)</td>
<td>35.9</td>
<td>36.6</td>
<td>68.8</td>
<td>42.3</td>
<td>68.1</td>
</tr>
<tr>
<td>T5-base (15b)</td>
<td>38.3</td>
<td>39.2</td>
<td>71.1</td>
<td>43.1</td>
<td>73.7</td>
</tr>
<tr>
<td>T5-large (15l)</td>
<td>39.4</td>
<td>40.3</td>
<td>72.0</td>
<td>42.9</td>
<td>73.0</td>
</tr>
<tr>
<td>T5-xl (15x)</td>
<td>39.6</td>
<td>40.6</td>
<td>71.8</td>
<td>42.2</td>
<td>74.6</td>
</tr>
</tbody>
</table>

**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. For ED2LM-base, its effectiveness is not signif-

---

5The 1st-3rd best runs (Qiao et al., 2021) in 2020 used TREC DL 2019 data for fine-tuning.
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-6L×2 provides 4X compression and ED2LM-F-8L×3 provides 8X compression by reducing the sequence length in the encoder. It can be seen that ED2LM-F-6L×2 outperforms T5-small and performs as well as BERT-base and PreTTR. Furthermore, while ED2LM-F-8L×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 MS MARCO benchmark.

### 5.2 Ablation 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 achieve 38.7 when trained on the “RQA-Merge” dataset. This is also true for PreTTR, which sees

<table>
<thead>
<tr>
<th>ED2LM →</th>
<th>Small</th>
<th>Base</th>
<th>Large</th>
<th>xl</th>
<th>F-6L×2</th>
<th>F-8L×3</th>
</tr>
</thead>
<tbody>
<tr>
<td>T5-small</td>
<td>F:4.4x/L:2.0x</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>F:2.4x/L:0.8x</td>
<td>r:<del>/n</del>/m/</td>
</tr>
<tr>
<td>PreTTR</td>
<td>F:5.2x/L:2.9x</td>
<td>r:<del>/n</del>/m/</td>
<td>F:1.2x/L:1.0x</td>
<td>-</td>
<td>F:2.9x/L:1.3x</td>
<td>r:<del>/n</del>/m/</td>
</tr>
<tr>
<td>BERT-base</td>
<td>F:10.4x/L:6.8x</td>
<td>r:<del>/n</del>/m/</td>
<td>F:2.5x/L:2.4x</td>
<td>-</td>
<td>F:5.8x/L:2.9x</td>
<td>r:<del>/n</del>/m/</td>
</tr>
<tr>
<td>T5-base</td>
<td>-</td>
<td>F:3.2x/L:2.3x</td>
<td>r:<del>/n</del>/m/</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>T5-large</td>
<td>-</td>
<td>F:9.6x/L:6.2x</td>
<td>r:<del>/n</del>/m/</td>
<td>F:2.8x/L:3.4x</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>T5-xl</td>
<td>-</td>
<td>-</td>
<td>F:10.3x/L:7.5x</td>
<td>r:<del>/n</del>/m/</td>
<td>F:2.6x/L:3.0x</td>
<td>r:<del>/n</del>/m/</td>
</tr>
</tbody>
</table>
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).

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

6 ED2LM for Question Generation

Question generation has played an important role for different downstream tasks (Shakeri et al., 2020; Puri et al., 2020; Del Tredici et al., 2021). We conjecture that the combination of generation and ranking loss used in ED2LM has the potential to improve question generation when compared to models trained with generation loss only. We evaluate this conjecture by comparing questions generated by vanilla generator trained with question likelihood only (Nogueira et al., 2019b) and ED2LM in different scenarios: manual inspection, assessment with automatic metrics and synthetic training data generation. For question generation task, an extra “eos” token (namely, the end of sequence token) is inserted between the question and the true/false token. Our pilot experiments show that this change does not influence the ranking performance but boosts the generation quality of ED2LM. We adopt the top-k sampling decoding (Fan et al., 2018) (set \(k = 10\)) in question generation for all models.

6.1 Question Generation with Less Hallucination

Manual inspection of the generated questions. We 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...
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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Question</th>
<th>Answerable ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>T5</td>
<td>is a modifier factor English</td>
<td>No</td>
</tr>
<tr>
<td>T5</td>
<td>what is experience modifier rating</td>
<td>No</td>
</tr>
<tr>
<td>ED2LM</td>
<td>what is an experience modifier in an insurance policy</td>
<td>Yes</td>
</tr>
<tr>
<td>ED2LM</td>
<td>experience modifier definition</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 4: Example generations from ED2LM-base and T5-base.

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.

**Question vs. paragraph overlap.** We 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 has an overlap rate of 55.62% (i.e., 55.62% of question tokens also appear in the source paragraph), ED2LM has an overlap rate of 62.14%, which is more than 6% higher than T5 model. In Table 4, we present example questions generated by T5 and ED2LM and their source paragraphs. Although T5 questions are somewhat related to the paragraph, the paragraph is not a good answer for them. For example, in Table 4, the first question T5 hallucinates the word English, which compromises the question quality.

### 6.2 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$_{\text{large}}$ dual encoder model using the synthetic question-passage pairs generated by ED2LM and T5 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 sampling passages from the same document. During training, we use both in-batch negatives and hard negatives. During inference, we retrieve top 1000 passages for each question from the passage collection containing about 8.8 million passages and report MRR@10. The model using ED2LM generated data achieves MRR@10=30.4, whereas the model using T5 generated data 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.

### 7 Conclusion

In this work, we propose a novel model named ED2LM. 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. We further demonstrate that ED2LM could generate questions with less hallucination. For future works, we plan to investigate the uses of ED2LM for different (generation) tasks such as multi-sentence compression (MRC) (Zhao et al., 2019), headline generation (Shen et al., 2019), and list question answering (Katti et al., 2021).
References


Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaojiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. RocketQA-v2: A Joint Training Method for Dense Passage Retrieval and Passage Re-ranking.


