Typos Correction Training Against Misspellings from Text-to-Text Transformers

Guicai Xie[1], **Ke Zhang**[1], **Lei Duan**[2], **Wei Zhang**[1], **Zeqian Huang**[1*]
[1]Machine Learning Platform Department, Tencent Inc., China
[2]School of Computer Science, Sichuan University, Chengdu, China
{blakexie, krezhang, ericwzhang, bobbyhuang}@tencent.com, leiduan@scu.edu.cn

Abstract

Dense retrieval (DR) has become a mainstream approach to information seeking, where a system is required to return relevant information to a user query. In real-life applications, typoed queries resulting from the users' mistyping words or phonetic typing errors exist widely in search behaviors. Current dense retrievers experience a significant drop in retrieval effectiveness when they encounter typoed queries. Therefore, the search system requires the extra introduction of spell-checkers to deal with typos and then applies the DR model to perform robust matching. Herein, we argue that directly conducting the typos correction training would be beneficial to make an end-to-end retriever against misspellings. To this end, we propose a novel approach that can facilitate the incorporation of the spelling correction objective into the DR model using the encoder-decoder architecture. During typos correction training, we also develop a prompt-based augmentation technique to enhance the DR space alignment of the typoed query and its original query. Extensive experiments demonstrate that the effectiveness of our proposed end-to-end retriever significantly outperforms existing typos-aware training approaches and sophisticated training advanced retrievers. Our code is available at https://github.com/striver314/ToCoTR.

Keywords: Information retrieval, Misspelled queries, Typos correction training

1. Introduction

Dense retrieval has been widely applied in a variety of applications, like web search (Mitra et al., 2017), question-answering (Karpukhin et al., 2020; Qu et al., 2021), and dialogue systems (Ji et al., 2014). It relies on the excellent text representation capabilities of pre-trained language models (PLMs) and learns dense representations to construct the DR model that follows a dual-encoder paradigm (Xiong et al., 2021; Karpukhin et al., 2020). In this typical architecture, the retriever first encodes queries and passages into a latent fixed-dimensional embedding space using two separate PLM-based encoders (e.g., BERT), then applies approximate nearest neighbor search (Johnson et al., 2021; Xiong et al., 2021) to efficiently retrieve these relevant passages given an input query.

In real-life search applications, typoed queries are frequent resulting from users' mistyping behaviors (Spink et al., 2001; Wilbur et al., 2006). Recent studies have been discussed where the DR model shows unexpectedly low effectiveness when confronted with queries containing typos (Zhuang and Zuccon, 2021, 2022; Sidiropoulos and Kanoulas, 2022; Chen et al., 2022). That is, even if the typos occur in a random token of a query, there will be a difference between the resulting typoed query embedding and the corresponding original query embedding, thus affecting the effectiveness of retrievers. As shown in Figure 1, typoed queries can lead to a significant drop in retrieval effectiveness even with these sophisticated training advanced retrievers like coCondenser (Gao and Callan, 2022) and SimLM (Wang et al., 2023).

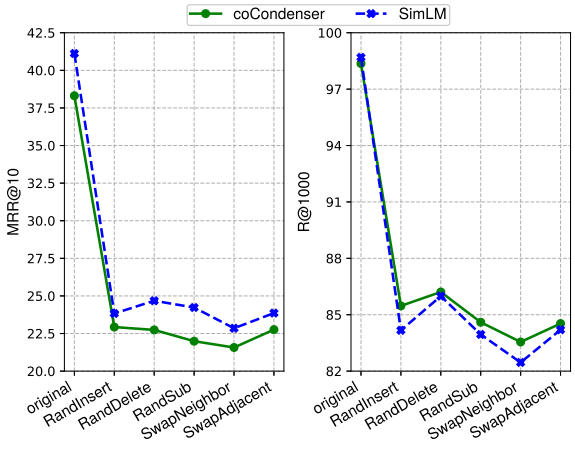


Figure 1: MRR@10 and R@1000 results on MS-MARCO. These five typos are detailed in Section 4.2. A significant drop in effectiveness across different types of simulated typos on queries.

Considering those out-of-domain typoed queries, the aforementioned works applied typos-aware training strategies to build typo-robust retrievers. Zhuang and Zuccon (2021) first investigated that typos can confuse BERT-based encoders for re-ranking and augmented the typoed queries into training queries, ensuring a mix of queries without

*Corresponding author

LREC-COLING 2024, pages 16907–16918

and with typos included in training queries. Later works use either contrastive learning or information distillation and alignment to keep the DR model robust to queries with typos. For example, combining data augmentation with contrastive learning to make the representation of typoed queries close to their original queries in the latent space (Sidiropoulos and Kanoulas, 2022), or utilizing distribution alignment mechanism to distill knowledge from original queries into typoed queries (Zhuang and Zuccon, 2022; Chen et al., 2022). These methods significantly improve the robustness of the DR model on typoed queries by implicitly aligning latent embedding, rather than directly conducting the spelling correction. However, their effectiveness on typoed queries is still lower than some sophisticated training advanced retrievers, or those combinations of utilizing advanced spell-checkers and DR retrievers. (Zhuang and Zuccon, 2022). This means that in the situation of queries with typos, the search system still needs to utilize the two-stage solution involving spell-checkers.

To build an end-to-end robust dense retriever, our idea is to incorporate spelling correction into the training pipeline of the standard DR model. In practice, spelling correction is formulated as a monolingual translation task and treated with an encoder-decoder based model (Junczys-Dowmunt et al., 2018; Yuan et al., 2021). Most current DR models have applied encoder-only PLMs as the backbone encoders, which is incompatible with the general architecture for spelling correction. On this basis, the challenge is how to effectively incorporate the spelling correction objective into the dual-encoder DR model so that relevant passages can be retrieved given typoed queries.

Towards this end, we present a novel approach to improve the robustness of dense text retrieval, which incorporates **T**yp**o**s **Co**rrection training into text-to-text transformer (**T**5) based **R**etrievers, called **ToCoTR**. This approach builds upon recent advances in T5 (Raffel et al., 2020) to learn text representations for dense retrieval (Ni et al., 2022a,b). The major contribution of ToCoTR is to provide an integrated strategy for DR training coupled with typos correction training. During the training pipeline of the DR, we additionally add typos correction training to achieve the alignment of misspelled words to original words. Furthermore, we develop a prompt-based augmentation technique to enhance the typos correction training and obtain a better representation of the typoed queries.

Briefly, our main contributions are as follows:

- We explore different incorporating strategies to conduct typos correction training. This establishes a feasible and effective approach that explicitly incorporates typos correction training into the training pipeline of dense retrieval.
- We propose a simple yet effective prompt-based augmentation technique to enhance the typos correction training. It adaptively realizes the alignment of typoed words to correct words and reduces the difference between typoed query embedding and its corresponding correct query embedding.
- We conduct a comprehensive comparison study to show the retrieval effectiveness of ToCoTR on queries with typos across three benchmark datasets.

2. Related Works

2.1. Robustness of Dense Retrievers

Traditional lexical matching methods or bag-of-words models (e.g., BM25 and TF-IDF) cannot solve these complex queries well due to the incapability of handling the term mismatch issue (Karpukhin et al., 2020; Xiong et al., 2021). Contrastively, the dual-encoder DR model can solve such problems by taking advantage of the powerful semantic matching capabilities of PLMs, such as BERT, RoBERTa, and ERNIE.

There are increasing concerns about the robustness of the DR model to typoed queries. Zhuang and Zuccon (2021) was the first study that shows typos occur in queries can lead to a significant drop in retrieval effectiveness and applies the data augmentation strategy to address the issue. Sidiropoulos and Kanoulas (2022) combined data augmentation with contrastive learning (CL) to bring the representation of an original query close to its typoed variations in the latent space while keeping it distant from other distinct queries. Furtherly, Zhuang and Zuccon (2022) provided insight into the DR model is sensitive to typoed queries caused by the BERT WordPiece tokenizer, and applied CharacterBERT instead of standard BERT as the backbone encoders of DRs. In this work, they also introduced a self-teaching (ST) strategy to reduce the difference between the relevance score distribution obtained from the typoed query and the relevance score distribution obtained from the corresponding original query (i.e., query without typos). Similarly, RoDR was proposed by Chen et al. (2022), which employs the same formulations to maintain the relative positions of query-passage pairs in the DR space. Meanwhile, hard negative mining was introduced by RoDR to train the final retriever.

Apart from considering the above situation of queries with typos, some works discuss the generalization ability of dense retrievers with different techniques. Edizel et al. (2019) proposed a method combining FastText with a supervised task to learn misspelling patterns and obtained word embeddings that are resilient to misspellings. Sentence-

T5 (Ni et al., 2022a) and GTR (Ni et al., 2022b) scale up the model capacity while keeping the bottleneck embedding size fixed. Notably, Sentence-T5 explores three architectures for extracting sentence representations from T5-style models and demonstrates that the sentence embedding capability of the encoder-decoder architecture is a promising technology. coCondenser (Gao and Callan, 2022) encodes all the important information of the given text to conduct robust matching in two-round retriever training manner. SimLM (Wang et al., 2023) adapt the underlying dense encoders for retrieval tasks with replaced language modeling objective in pre-training architecture, which does not have skip connections between encoder and decoder.

2.2. Spelling Correction Task

Spelling correction is the task of correcting spell errors or misspellings of words in a given text. In recent years, transformer-based spelling correction approaches had become a dominant paradigm, including sequence-to-edit (Seq2Edit) (Omelianchuk et al., 2020; Gu et al., 2019; Awasthi et al., 2019) and sequence-to-sequence (Seq2Seq) (Rothe et al., 2021; Stahlberg and Kumar, 2021). Seq2Edit-based approaches achieve correction by predicting a sequence of edit operations, while Seq2Seq-based approaches are able to condition the detection and correction of the input text as if it were text generation. Considering the flexibility and effectiveness of Seq2Seq-based methods, we would follow the previous work and utilize the teacher forcing negative log-likelihood loss to conduct typos correction training.

3. Incorporating Typos Correction Training for T5 Retriever

In this section, we describe a novel approach for incorporating typos correction training into the dual-encoder dense retrieval (called **ToCoTR**).

3.1. T5 Dual-encoder Architecture

In order to conveniently facilitate typos correction training, the backbone encoder of the DR model is preferably an encoder-decoder architecture and requires keeping the powerful representation capability. As it happened, encoder-decoder sentence embedding models have proved to be a promising architecture (Ni et al., 2022a).

Technically, we directly use the off-the-shelf pre-trained T5-style model as the backbone encoder to train the dense retrieval in a dual-encoder paradigm. Compared with the encoder-only model, the T5-style model does not place a special symbol (e.g., `[CLS]` in BERT) at the beginning of the text sequence. We follow prior work (Ni et al., 2022a) and assume that the decoder generates the first token prediction containing the semantics of the entire input text sequence, so it is straightforward to use the first token prediction of the decoder as the representations of query or passage. The architecture of the T5 dual-encoder is illustrated in Figure 2.

Formally, let q denote a query and p_i denote passage from a large set $D = \{p_i\}_{i=1}^{m}$ of m passages. Given a query, passage retrieval aims to return a sorted list of the n most relevant passages $L = [p_1, p_2, ..., p_n]$ according to the relevance score of the retrieval model. In the phase of training DR, we assume a set of binary positive correlation judgments as supervised signals, denoted by $R = \langle q_i, p_i^+, \{p_{i,1}^-, p_{i,2}^-, ..., p_{i,s}^-\}\rangle$, where p_i^+ denotes the relevant passages and p_i^- denotes the irrelevant passages for query q_i. To optimize the dense retriever, the negative log-likelihood (NLL) loss is applied as follows:

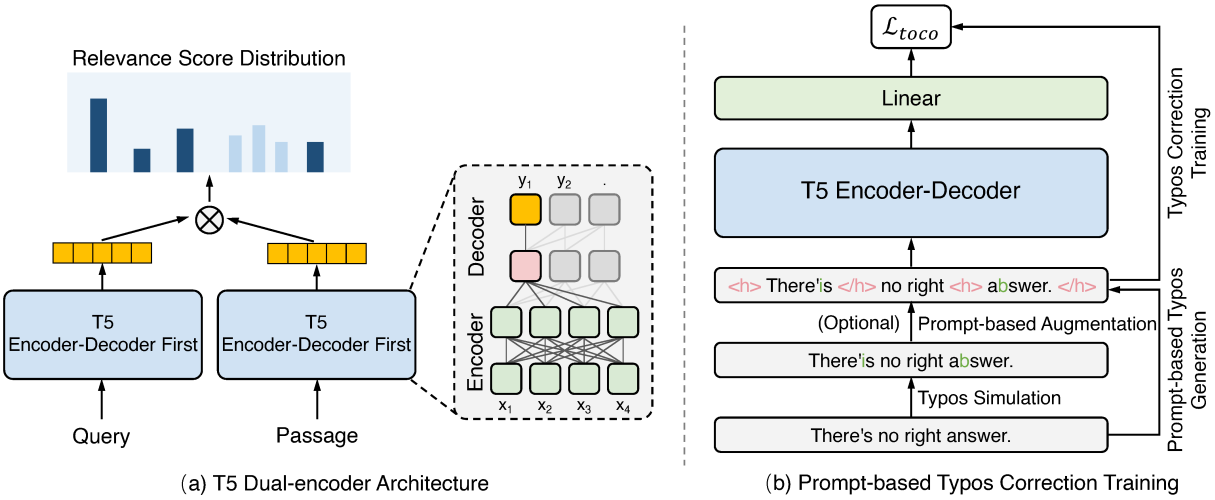
$$\begin{aligned}&\mathcal{L}_{nll}\left(\langle q_i, p_i^+, \{p_{i,1}^-, p_{i,2}^-, ..., p_{i,s}^-\}\rangle\right)\\&= -\log \frac{e^{sim(q_i, p_i^+)}}{e^{sim(q_i, p_i^+)} + \sum_{j=1}^{s} e^{sim(q_i, p_{i,j}^-)}}\end{aligned} \tag{1}$$

3.2. Prompt-based Typos Correction Training

To enable the ToCoTR to have the ability of textual spelling correction and explicitly achieve the alignment of misspelled words to correct words in the DR latent space, we introduce the prompt-based typos correction training into the DR model training pipeline. Figure 2 depicts the prompt-based typos correction training framework. It consists of two components:

- A prompt-based typos generation module that simulates different misspelled sentences for input samples at the token embedding layer.
- A typos correction training module that computes sentence representations of each text and optimizes the error correction objective. During training, we use the average of decoder outputs across all input tokens.

The spelling correction task is treated as a Seq2Seq task. When given a source text $X = (x_1, x_2, ..., x_t)$, we first pass it to the prompt-based typos generation module, in which five generators (see Section 4.2 for details) are applied to generate another variation X^{typo}. After that, the model takes X^{typo} as input and outputs a target text $Y = (y_1, y_2, ..., y_s)$. The task can be formulated as a conditional generation problem by modeling and maximizing the conditional probability $\mathbf{P}(Y \mid X)$.



(a) T5 Dual-encoder Architecture (b) Prompt-based Typos Correction Training

Figure 2: Illustration of the T5 dual-encoder architecture and the framework of prompt-based typos correction training respectively. For prompt-based typos correction training, we use off-the-shelf pre-trained models such as T5_{base}. The symbols <h> and </h> denote the extra tokens from prompt-based augmentation. The case is sourced MSMARCO corpus.

During typos generation, given a set of context text $C = \{X_1, X_2, ..., X_{|C|}\}$, typoed text X^{typo} are simulated based on the rule of injecting typos into the source text. As follows, we first randomly select 80% of the texts from C to construct typoed texts, and the remaining 20% are correct texts, i.e. texts without conducting the typos simulation[1]. Then for each selected source text $X_k \in C$, we choose $\alpha\,|X_k|$ token positions at random to simulate typos, where α denotes the proportion. If the t-th token is chosen then use a randomly selected typos generator to inject the typos.

Additionally, we are inspired by Chan and Fan (2019) and propose the prompt-based typos augmentation technique to enhance the ability of alignment between typoed tokens and correct tokens. Specifically, the prompt-based typos augmentation technique is the optional solution, which adds symbols <h> and </h> to each side of the typoed word after it has been injected. This augmentation can be briefly understood as the augmentation template "<h> [X] </h>", where [X] is a placeholder to put the misspelled word of the input text.

To allow the model to recover the typoed tokens conditioned on texts with typos, bringing the token embedding representation of the incorrect word closer to the correct word. We use off-the-shelf encoder-decoder based pre-trained models such as T5_{base} to initialize the model and then continue training on typos simulation training data. Following Cao et al. (2023), we adopt the teacher forcing negative log-likelihood loss as the objective and minimize the loss for a set of $|C|$ text pairs $\{\langle X_k^{typo}, Y_k\rangle\}_{k=1}^{|C|}$, as follows:

$$\mathcal{L}_{toco} = -\log \sum_{k=1}^{|C|} (\mathbf{P}(Y_k \mid X_k^{typo}; \theta)) \qquad (2)$$

where θ is the learning parameters of the model in typos correction training.

3.3. Training Procedure

The training strategy of incorporating typos correction training into the DR model is an important factor in methodology design. We propose two ways for incorporating typos correction training:

- **Joint training (joint)**. We jointly train the model with the NLL loss $\mathcal{L}_{nll}$ and typos correction loss $\mathcal{L}_{toco}$ in $\mathcal{L}_{joint} = \mathcal{L}_{nll} + \beta\mathcal{L}_{toco}$. β is a hyper-parameter to balance two objectives.
- **Typos correction training then DR transfer (two-stage)**. We first train the spelling correction objective in a single T5-based model using $\mathcal{L}_{toco}$, then train the dual-encoder retriever with $\mathcal{L}_{nll}$. The key is to initialize the backbone encoders of the DR model with the trained model parameters after finishing the typos correction training.

Besides typos correction training transfer, our approach can also be incorporated with existing training techniques to improve the robustness of retrievers. Two direct techniques that do not introduce extra training phrases are as follows:

Hard Negatives Mining. We directly employ the open-sourced hard negatives with mined by Rock-

[1]The sampling rate setting is consistent with the 80% [MASK] tokens used for word replacement during masked LM training.

etQAv2 (Ren et al., 2021) to optimize the retriever's representation in the training pipeline.

Coupled with Self-Teaching Training. The self-teaching technique (Zhuang and Zuccon, 2022) minimizes the KL-divergence between the passage match relevance distribution obtained by the typoed query q^{typo} and the distribution obtained from the corresponding correct query q in the formula $\mathcal{L}_{\mathrm{KL}}(s_{q^{typo}}, s_q) = s_{q^{typo}}(q^{typo}, p) \cdot \log \frac{s_{q^{typo}}(q^{typo},p)}{s_q(q,p)}$, where $s_q(q,p)$ and $s_{q^{typo}}(q^{typo},p)$ denote the normalized similarity score of q and q^{typo} associated with its passages respectively. During the retriever training, the final loss function are $\mathcal{L} = \mathcal{L}_{joint} + \mathcal{L}_{\mathrm{KL}}$ for joint training, and $\mathcal{L} = \mathcal{L}_{nll} + \mathcal{L}_{\mathrm{KL}}$ for two-stage training.

4. Experimental Setup

4.1. Datasets and Metrics

We adopt three public datasets in our experiment: including MS MARCO v1 passage ranking (MSMARCO) (Nguyen et al., 2016), the TREC Deep Learning Track Passage Retrieval Task 2019 (TREC 2019) (Craswell et al., 2020), and ANTIQUE (Hashemi et al., 2020; Penha et al., 2022). For misspelled queries, the first two data are simulated typoed queries, and the latter one is manually validated typoed queries. Table 1 shows the statistics of the three datasets.

MSMARCO dataset contains a large number of queries sampled from Bing search logs and annotated with binary relevant passages in web documents. TREC 2019 dataset uses the same passages corpus as MSMARCO but differs in terms of queries with four-level relevance annotations, ranging from 0 (irrelevant) to 3 (perfectly relevant). The shared data collection consists of 8.8 million passages. These two aforementioned datasets will be used to generate typoed queries via five typos generators (repeat 10 times).

Besides, manually validated typoed queries released by Penha et al. (2022) are also used. ANTIQUE contains manual four-level relevance annotations (from 1 to 4) and collects 2,626 non-factual questions and 403,666 answers from a community answering service. There are three released query variations on misspellings[2], which are used to evaluate the DR model in the zero-shot setting. The average results of the three variations are reported in the experiments.

Following previous work, we adopt the metrics commonly used in each dataset for evaluation. For TREC 2019, these Normalized Discounted Cumulative Gain at 10 (nDCG@10), Mean Reciprocal Rank (MRR), and Mean Average Precision (MAP) are considered. For MSMARCO dev queries, the MRR at 10 (MRR@10) and Recall at top 1000 ranks (R@1000) to evaluate the performance of passage retrieval. For the ANTIQUE dataset, we follow RoDR (Chen et al., 2022) using nDCG@10 and R@1000 metrics and follow the data creators Hashemi et al. (2020) using MAP as a recall-oriented metric. Considering that TREC 2019 and ANTIQUE are four-level relevance scores, the first two levels are mapped as relevant and the last two levels are mapped as irrelevant to calculate binary metrics (e.g., MAP, MRR) and map the labels as 0 to 3 to compute nDCG@10.

We employ `ranx` evaluation library (Bassani, 2022) to measure performance and conducted a two-tailed paired t-test with Bonferroni correction to measure the statistical significance ($p < 0.05$).

Dataset	Train	Dev	Test	Avg. q length
MSMARCO	400,782	6,980	6,837	6.10
TREC 2019§	-	-	43	5.60
ANTIQUE	2,426	-	200	10.82

Table 1: Detailed statistics for the experimental datasets, including the average length of queries. Dataset marked with § indicates that only judged queries are given.

4.2. Typos Simulation

In this section, we introduce five typos generators that simulated generated different misspelled typos. The details are described below:

- **Rand**-(**Insert**, **Delete**, **Sub**): Randomly inserts, deletes, or substitutes a random character within a randomly chosen word, e.g., `typo` $\rightarrow$ {`typos`, `typ`, `type`}.
- **SwapNeighbor**: Randomly swaps a character with one of its neighbor characters, e.g., `typo` $\rightarrow$ `tyop`.
- **SwapAdjacent**: Randomly swaps a character with one of its adjacent letter on the QWERTY keyboard, e.g., `typo` $\rightarrow$ `typp`.

4.3. Methods for Comparison

To have a comprehensive comparison, we choose as baselines the state-of-the-art methods that consider both open-sourced retrievers and typos-aware dense retrievers. The methods in our comparative evaluation are the follows.

The open-sourced retrievers include the traditional sparse retriever BM25 (Robertson and Jones, 1976; Robertson and Zaragoza, 2009), and dense

[2]NeighbCharSwap, QWERTYCharSub, RandomCharSub are equivalent to SwapNeighbor, SwapAdjacent, RandSub respectively.

retrievers DPR (Karpukhin et al., 2020), CharacterBERT (Boukkouri et al., 2020), Sentence-T5 (Ni et al., 2022a), coCondenser (Gao and Callan, 2022), SimLM (Wang et al., 2023).

The typos-aware dense retrievers include BERT+Aug (Zhuang and Zuccon, 2021), BERT+Aug+CL (Sidiropoulos and Kanoulas, 2022), CharacterBERT+ST (Zhuang and Zuccon, 2022) and RoDR (Chen et al., 2022).

4.4. Implementation Details

Among all baselines, we use the Pyserini (Lin et al., 2021) implementation to produce baseline BM25 results on all datasets. Since coCondenser and SimLM do not report retrieval results on misspelled queries, we use open-source retrievers to produce the results. Besides, BERT+Aug+CL is our implementation because its code was not available when we conducted this work. Other baselines can be directly used in corresponding implementations for comparison.

Our model training is based on the Tevatron DR training toolkit (Gao et al., 2022) on a single NVIDIA Tesla V100 GPU with 32G RAM. For the typos correction training of our model, we utilize the start checkpoint t5-base[3] to initialize the model parameters θ, and utilize the pre-processed passages corpus of MSMARCO as the train data. We use the open-source library `TextAttack` (Morris et al., 2020) to simulate typos in the training phrase, and the proportion of typos per input text α is set as 0.2 (see Figure 3).

For the DR model training, we use the AdamW optimizer with a learning rate of 5e-5. The model is trained up to 150,000 steps on a linear learning rate schedule with a batch size of 8. The max query length is 32, and the max passage length is 128. The ratio of the positive to the hard negative is set to 1:7. The 7 negative passages are randomly sampled from the top 200 passages which are retrieved by RocketQAv2 and in-batch negatives sampling is applied to each training sample.

5. Results and Analysis

In this section, we first describe the comparing results, then compare ToCoTR with an alternative two-stage search engine architecture that applies different spell-checkers to correct spelling errors before a query is retrieved. Finally, we present a detailed analysis of ToCoTR, including an ablation study.

[3]https://huggingface.co/t5-base

5.1. Main Results

The retrieval results on both queries without and with typos are shown in Table 2. It can be observed that:

Considering methods that do not deal with typos in queries (runs a-f in Table 2) and their retrieval effectiveness on queries without typos, all dense retrievers outperform the BM25, which is consistent with the conclusion of the previous study (Zhuang and Zuccon, 2021). coCondenser and SimLM are more significantly effective than DPR, CharacterBERT, and Sentence-T5 because they undergo sophisticated, computationally expensive, multi-stage training and apply some training techniques such as the gradient cache technique (Gao et al., 2021).

We also compare the results on queries without typos between Sentence-T5 and ToCoTR, which use the same start checkpoint (T5$_{base}$) and the same backbone encoder to encoder queries and passages. The results show that ToCoTR achieves higher effectiveness among three datasets. This improvement may be due to the training strategy or combination of training strategies used by ToCoTR. We will discuss this in detail in Section 5.3.

Then turn our attention to the typos-aware dense retrievers (runs g-k): these methods are trained with exactly the same setting and the only difference is applied different typos-aware training strategies. These results describe that, on queries without typos, ToCoTR has better effectiveness across different metrics and datasets, with the statistically significant difference obtained on both MRR@10 and R@1000 for MSMARCO, and MAP for TREC 2019.

Next, we discuss the results of queries with typos among runs l-q in Table 2. All retrievers that are not pertinently designed to tackle queries with typos return results that are obviously lower than their corresponding queries without typos (runs a-f). Oppositely, those dense retrievers trained using typos-aware strategies (runs r-v) perform a smaller average drop rate compared to the aforementioned methods. Overall, among all methods confronting queries with typos (runs l-v), the ToCoTR dramatically outperforms the other methods and is significantly different from almost all other methods, including coCondenser and SimLM. This can fully demonstrate the advantages of typos correction training.

We further notice that both the coCondenser and SimLM methods are competitive with other methods that conduct typos-aware training (rather than explicit spelling correction) and are only lower than the CharacterBERT+ST on both TREC 2019 and MSMARCO datasets. These results show that sophisticated training mechanisms can also improve the robustness of the DR model on queries with typos. Similarly, it is expected that the adaptation

Queries	Methods	PLMs	TREC 2019			MSMARCO		ANTIQUE		
			nDCG@10	MRR	MAP	MRR@10	R@1000	nDCG@10	MAP	R@1000
w/o typos	a) BM25 (pyserini)	-	50.6	70.4	30.1	18.7	85.7	23.7	15.9	46.1
	b) DPR	BERT_{base}	59.7	72.5	35.2	32.6	95.2	26.5	19.0	57.9
	c) CharacterBERT	CharacterBERT	61.6	78.5	33.0	32.1	94.8	25.4	17.9	55.2
	d) Sentence-T5	T5_{base}	64.3	82.4	36.4	32.1	95.9	30.0	22.1	61.8
	e) coCondenser†	Condenser	**71.5**	86.8	45.3	38.3	98.4	**32.4**‡	**24.4**‡	**66.9**‡
	f) SimLM†	BERT_{base}	71.4	**87.9**	**46.9**	**41.1**‡	**98.7**‡	-	-	-
	g) BERT+Aug	BERT_{base}	61.8	80.7	35.3	32.7	95.1	26.6	19.6	59.7
	h) BERT+Aug+CL	BERT_{base}	61.1	77.1	34.5	32.8	94.8	26.7	19.5	58.5
	i) CharacterBERT+ST	CharacterBERT	63.9	80.7	33.6	32.6	94.6	26.6	19.0	53.8
	j) RoDR	BERT_{base}	62.1	78.8	34.6	32.8	95.1	26.1	19.0	59.7
	k) ToCoTR (two-stage)	T5_{base}	**69.4**	**87.8**	**41.0**‡	**34.4**‡	**96.6**‡	**28.6**	**20.1**	**60.5**
w/ typos	l) BM25 (pyserini)	-	25.9	35.3	15.0	9.5	61.1	16.9	11.3	38.0
	m) DPR	BERT_{base}	28.4	41.3	15.9	13.5	68.2	16.4	11.4	42.9
	m) CharacterBERT	CharacterBERT	35.9	53.5	18.9	16.0	72.2	17.2	12.4	42.7
	o) Sentence-T5	T5_{base}	40.5	56.4	21.8	18.5	80.6	20.8	15.0	52.3
	p) coCondenser†	Condenser	**46.1**	**60.2**	27.6	22.1	**84.5**‡	**24.5**‡	**18.1**‡	**56.2**‡
	q) SimLM†	BERT_{base}	45.9	**60.2**	**27.7**	**23.6**‡	83.8	-	-	-
	r) BERT+Aug	BERT_{base}	42.9	59.4	23.8	21.8	84.2	20.6	14.8	50.2
	s) BERT+Aug+CL	BERT_{base}	43.9	59.8	24.1	22.9	85.6	21.0	15.1	49.7
	t) CharacterBERT+ST	CharacterBERT	52.0	70.2	27.0	26.4	89.2	22.2	15.8	48.0
	u) RoDR	BERT_{base}	43.9	59.0	23.9	23.2	86.2	21.0	15.0	51.0
	v) ToCoTR (two-stage)	T5_{base}	**63.4**‡	**83.5**	**36.6**‡	**31.3**‡	**94.6**‡	**26.1**‡	**18.6**‡	**57.1**‡

Table 2: Retrieval results for queries without typos and queries with typos, respectively. Results on queries containing typos are averaged by repeating the typo simulation procedure 10 times on MSMARCO and TREC 2019 (statistical significance computation from the first average) and averaged by three released query variations on ANTIQUE (statistical significance computation from the NeighbCharSwap average). Results with † are from our reproduction with open-source model checkpoints. The best result and the second-best score are in bold and underlined font, respectively. We use ‡ indicate significant differences at p-value < 0.05.

Methods	w/o typos		w/ typos	
	MRR@10	R@1000	MRR@10	R@1000
pyspellchecker → CharacterBERT	27.3	88.5	23.0	81.9
MS-Spellchecker → CharacterBERT	32.0	94.6	29.9	91.3
GG-Spellchecker → CharacterBERT	32.2	94.8	29.4	90.4
pyspellchecker → Sentence-T5	28.4	91.9	24.7	87.0
MS-Spellchecker → Sentence-T5	32.0	95.9	30.5	93.6
GG-Spellchecker → Sentence-T5	32.0	95.9	29.9	93.1
ToCoTR (two-stage)	**34.4**‡	**96.6**‡	**31.3**	**94.6**‡

Table 3: Comparison among ToCoTR and CharacterBERT, Sentence-T5 involving alternative three spell-checkers on MSMARCO. Statistically significant better than others at p-value < 0.05 are marked with ‡.

of ToCoTR to the same training mechanisms of coCondenser and SimLM would further improve its effectiveness, including the situation of queries without typos. We will confirm this hypothesis in future work.

5.2. Comparison with the DR Model involving Spell-checkers

In this part, we compare ToCoTR with a common search engine architecture involving spell-checkers. In this architecture, a spell-checker is used to detect and correct typos in the pre-processing of queries and then to conduct relevant passage retrieval. Some queries without typos may be incorrectly detected and corrected, which is related to the capability of the spell-checker. The previous study (Zhuang and Zuccon, 2022) highlighted that the current state-of-the-art typos-aware training method returns lower effectiveness than the corresponding solution involving spell-checkers. Furtherly, we employ three spell checkers in combination with CharacterBERT and Sentence-T5 for our experiments, including pyspellchecker[4], which is a rule-based spell-checking toolkit that relies on dictionary-based rule sets, Microsoft Bing Spell Check API[5] (MS-Spellchecker), which utilizes machine learning and statistical machine translation to provide accurate and contextual corrections, and Google Search API[6] (GG-Spellchecker), which has been shown in previous research (Hagen et al., 2017) to be possibly the most useful spell corrections.

Table 3 compares ToCoTR with the pipelines of three different spell-checkers in combination with the CharacterBERT and Sentence-T5 dense retrieval models on MSMARCO. Clearly, the ToCoTR retriever significantly outperforms all six combination solutions that involve spell-checkers, demonstrating that the retriever benefits from our pro-

[4]https://github.com/barrust/pyspellchecker

[5]https://learn.microsoft.com/en-us/azure/cognitive-services/bing-spell-check/overview

[6]https://developers.google.com/custom-search/v1/introduction

Methods	w/o typos		w/ typos	
	MRR@10	R@1000	MRR@10	R@1000
ToCoTR	34.4	96.6	31.3	94.6
w/o Hard Negative	33.5$^{\downarrow}$	95.7$^{\downarrow}$	29.8$^{\downarrow}$	93.2$^{\downarrow}$
w/o Self-Teaching	34.1	97.0	27.3$^{\downarrow}$	91.9$^{\downarrow}$
w/o Prompt	34.1	96.7	30.6$^{\downarrow}$	94.3
w/o ToCo	33.9	96.6	28.7$^{\downarrow}$	92.4$^{\downarrow}$

Table 4: The results of different variants of ToCoTR retriever on MSMARCO. Statistically significant drops at p-value < 0.05 are marked with $\downarrow$.

Methods	w/o typos		w/ typos	
	MRR@10	R@1000	MRR@10	R@1000
Two-stage	**33.5**‡	**95.7**	**29.8**‡	**93.2**‡
Joint	32.8	**95.7**	26.8	90.8

Table 5: The results of two different ways for incorporating typos correction training on MSMARCO. The hyper-parameter β of balance $\mathcal{L}_{nll}$ and $\mathcal{L}_{toco}$ is set as 0.1 in joint training. Statistically significant differences at p-value < 0.05 are marked with $\ddagger$.

posed prompt-based typos correction training strategy, which has the capacity to implicitly correct typos in queries and align the typoed word to the correct word. This means that the end-to-end ToCoTR retriever incorporating the typos correction training is a promising approach.

5.3. Ablation Study

A detailed ablation study is conducted to demonstrate the impact of the different training stages in ToCoTR design. Specifically, we remove one component at a time from the ToCoTR. First, we name ToCoTR without the different components as follows:

- **w/o Hard Negative**: ToCoTR without hard negative mining, which only apply BM25 hard negatives to train retriever using $\mathcal{L}_{nll}$, instead of using open-sourced hard negatives mined from RocketQAv2.
- **w/o Self-Teaching**: ToCoTR without coupled with self-teaching training, that is the final training loss does not include $\mathcal{L}_{\mathrm{KL}}$ during the training retrieval.
- **w/o Prompt**: ToCoTR without prompt-based augmentation during typos correction training, i.e. the misspelled words are not augmented with the symbols <h> and </h>.
- **w/o ToCo**: ToCoTR without introducing the typos correction training, which means that the retriever is trained without using loss $\mathcal{L}_{toco}$. This is equivalent to directly replacing the backbone encoders with the T5 model and then using the encoder-decoder first architecture to train the retriever.

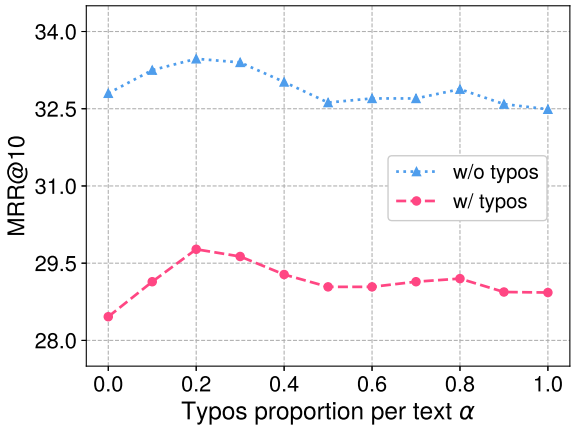


Figure 3: MRR@10 results of passage retrieval of both queries with and without typos with different typos proportion of per input text α in typos correction training on MSMARCO.

Results measured using MRR@10 and R@1000 on MSMARCO are shown in Table 4. Several observations from the results are worth highlighting: (1) The contribution of the prompt-based augmentation technique (namely, w/o Prompt) is not strikingly high, but it is still beneficial to the typoed queries while keeping a statistically significant difference on MRR@10. (2) Without typos correction training (namely, w/o ToCo), the performance of ToCOTR on both original and typoed queries is greatly affected. (3) Hard negatives mining (namely, w/o Hard Negative) and self-teaching strategy (namely, w/o Self-Teaching) are also important in ToCoTR since their removal leads to marked effectiveness loss. Moreover, the adaption of hard negative mining can demonstrate why advanced retrievers with complex training can outperform typo-aware training retrievers such as BERT-Aug and RoDR, as discussed in Section 5.1.

5.4. Analysis on Typos Correction Training

In this section, we analyze the retrieval results by incorporating the typos correction training in different training ways. Then we conduct a detailed analysis of the injected typos proportion per input text in typos correction training.

Joint or Two-stage? To examine the effect of typos correction training ways, we utilize the T5$_{base}$ as the backbone encoders of dual-encoders coupled with self-teaching training to train a retriever, as described in Section 3.3. Table 5 shows the performance drop on queries with typos in terms of MRR@10 and R@1000 in the joint training way,

indicating that the appropriate way of incorporating typos correction training into the DR model is two-stage training. The reason is that since the encoder-decoder first architecture only uses the embedding of the first token as the whole sentence representation, while spelling errors can occur anywhere in the sentence, utilizing the representation of the first token for conducting typos correction training is inadequate. Besides, this finding provides guidance to adapt typos correction training to sophisticated training mechanisms like coCondenser and SimLM.

The Impact of the Number of Typos. During typos correction training, we focus on the impact of the number of injecting typos in individual input text. As we described in the previous section, for each text, randomly selecting words to be injected with typos in different predetermined values (equivalent to the hyperparameter α) to validate the contribution of the typos correction training to train a robust DR model. Figure 3 shows the impact of the number of typos on both queries with and without typos. Clearly, we can observe that the best MRR@10 metric is achieved for the typos proportion per input text of 0.2.

6. Conclusion and Future Works

This paper has presented a novel approach to improve the robustness of retrievers by incorporating typos correction training into the DR model in a two-stage manner. To integrate the spelling correction task, we incorporate the spelling correction objective into the training pipeline of the dual-encoder DR model using encoder-decoder architecture. We also implement a prompt-based augmentation technique to enhance the typos correction training. Extensive results demonstrate the effectiveness of ToCoTR, where the incorporation of typos error correction training into dense retrievers outperforms those with typos-aware training retrievers, and even outperforms the combining solutions involving advanced spell checkers for DR models. This helps drive the use of end-to-end dense retrievers in search engine systems.

For future work, we plan to integrate typos correction training with the multi-stage training process of existing advanced retrievers to improve the retrieval effect of DR models when these models encounter misspelled texts. Further, we will also consider training typos correction training in an encoder-only architecture.

7. Limitations

One limitation of ToCoTR is that it cannot be used in the common encoder-only architecture as the backbone encoder for DRs, since the typos correction training is modeled as a Seq2Seq task with an encoder-decoder architecture. Because of this, the query latency of encoder-decoder retrievers is higher than encoder-only retrievers (as shown in Table 6, and the query latency on the GPU of ToCoTR is one millisecond lower than that of DPR). On the other hand, although the typos correction training of ToCoTR is quite efficient on queries with typos, it still requires extra computational costs to conduct the typos correction training in a two-stage manner.

8. Acknowledgements

We thank Wenchuan Yang, Xinye Wang and Yuening Qu for helpful discussions, and anonymous reviewers for their insightful comments.

9. Bibliographical References

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. 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 2019, Hong Kong, China*, pages 4259–4269. Association for Computational Linguistics.

Elias Bassani. 2022. ranx: A blazing-fast python library for ranking evaluation and comparison. In *Advances in Information Retrieval - 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway*, volume 13186 of *Lecture Notes in Computer Science*, pages 259–264. Springer.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 2020. Characterbert: Reconciling elmo and BERT for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online)*, pages 6903–6915. International Committee on Computational Linguistics.

Hannan Cao, Wenmian Yang, and Hwee Tou Ng. 2023. Mitigating exposure bias in grammatical error correction with data augmentation and reweighting. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023,*

Dubrovnik, Croatia., pages 2115–2127. Association for Computational Linguistics.

Ying-Hong Chan and Yao-Chung Fan. 2019. A recurrent BERT-based model for question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering, MRQA@EMNLP 2019, Hong Kong, China*, pages 154–162. Association for Computational Linguistics.

Xuanang Chen, Jian Luo, Ben He, Le Sun, and Yingfei Sun. 2022. Towards robust dense retrieval via local ranking alignment. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria*, pages 1980–1986. ijcai.org.

Bora Edizel, Aleksandra Piktus, Piotr Bojanowski, Rui Ferreira, Edouard Grave, and Fabrizio Silvestri. 2019. Misspelling oblivious word embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA*, pages 3226–3234. Association for Computational Linguistics.

Luyu Gao and Jamie Callan. 2022. Unsupervised corpus aware language model pre-training for dense passage retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, ACL 2022, Dublin, Ireland*, pages 2843–2853. Association for Computational Linguistics.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Tevatron: An efficient and flexible toolkit for dense retrieval. *CoRR*, abs/2203.05765.

Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. 2021. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP, RepL4NLP@ACL-IJCNLP 2021, Online*, pages 316–321. Association for Computational Linguistics.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Proceedings of the Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pages 11179–11189.

Matthias Hagen, Martin Potthast, Marcel Gohsen, Anja Rathgeber, and Benno Stein. 2017. A large-scale query spelling correction corpus. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan*, pages 1261–1264. ACM.

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *CoRR*, abs/1408.6988.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, Volume 1*, pages 595–606. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. 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 2020, Online*, pages 6769–6781. Association for Computational Linguistics.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Frassetto Nogueira. 2021. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada*, pages 2356–2362. ACM.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia*, pages 1291–1299. ACM.

John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online*, pages 119–126. Association for Computational Linguistics.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2022a. Sentence-T5: Scalable sentence

encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland*, pages 1864–1874. Association for Computational Linguistics.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Abrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. 2022b. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates*, pages 9844–9855. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem N. Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR - grammatical error correction: Tag, not rewrite. In *Proceedings of the 15th Workshop on Innovative Use of NLP for Building Educational Applications, BEA@ACL 2020, Online*, pages 163–170. Association for Computational Linguistics.

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, NAACL-HLT 2021, Online*, pages 5835–5847. Association for Computational Linguistics.

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*, 21:140:1–140:67.

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. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic*, pages 2825–2835. Association for Computational Linguistics.

Stephen E. Robertson and Karen Spärck Jones. 1976. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A simple recipe for multilingual grammatical error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, Virtual Event, August 1-6, 2021*, pages 702–707. Association for Computational Linguistics.

Georgios Sidiropoulos and Evangelos Kanoulas. 2022. Analysing the robustness of dual encoders for dense retrieval against misspellings. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '22, Madrid, Spain*, pages 2132–2136. ACM.

Amanda Spink, Dietmar Wolfram, Major B. J. Jansen, and Tefko Saracevic. 2001. Searching the web: The public and their queries. *Journal of the American Society for Information Science and Technology*, 52(3):226–234.

Felix Stahlberg and Shankar Kumar. 2021. Synthetic data generation for grammatical error correction with tagged corruption models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications, BEA@EACL, Online*, pages 37–47. Association for Computational Linguistics.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2023. Simlm: Pre-training with representation bottleneck for dense passage retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, ACL 2023, Toronto, Canada*, pages 2244–2258. Association for Computational Linguistics.

W. John Wilbur, Won Kim, and Natalie Xie. 2006. Spelling correction in the PubMed search engine. *Information Retrieval*, 9(5):543–564.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria*. OpenReview.net.

Zheng Yuan, Shiva Taslimipoor, Christopher Davis, and Christopher Bryant. 2021. Multi-class grammatical error detection for correction: A tale of two systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic*, pages 8722–8736. Association for Computational Linguistics.

Shengyao Zhuang and Guido Zuccon. 2021. Dealing with typos for BERT-based passage retrieval and ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic*, pages 2836–2842. Association for Computational Linguistics.

Shengyao Zhuang and Guido Zuccon. 2022. Characterbert and self-teaching for improving the robustness of dense retrievers on queries with typos. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '22, Madrid, Spain*, pages 1444–1454. ACM.

10. Language Resource References

Nick Craswell and Bhaskar Mitra and Emine Yilmaz and Daniel Campos and Ellen M. Voorhees. 2020. *Overview of the TREC 2019 deep learning track*. Microsoft AI & Research, University College London, and National Institute of Standards and Technology (NIST). PID https://github.com/ielab/CharacterBERT-DR/tree/main/data/dl2019.

Helia Hashemi and Mohammad Aliannejadi and Hamed Zamani and W. Bruce Croft. 2020. *ANTIQUE: A Non-factoid Question Answering Benchmark*. Center for Intelligent Information Retrieval at University of Massachusetts Amherst, and Information and Language Processing Systems (ILPS) at University of Amsterdam. PID https://ir-datasets.com/antique.html.

Tri Nguyen and Mir Rosenberg and Xia Song and Jianfeng Gao and Saurabh Tiwary and Rangan Majumder and Li Deng. 2016. *MS MARCO: A Human Generated MAchine Reading COmprehension Dataset*. Microsoft AI & Research. PID https://huggingface.co/datasets/Tevatron/msmarco-passage.

Gustavo Penha and Arthur Câmara and Claudia Hauff. 2022. *Evaluating the Robustness of Retrieval Pipelines with Query Variation Generators*. Delft University of Technology. PID https://github.com/Guzpenha/query_variation_generators.

A. Latency analysis

	API call	Query encoding	Index search
MS-Spellchecker	271.3ms	-	-
GG-Spellchecker*	437.2ms	-	-
DPR	-	0.2ms	226.1ms
CharacterBERT	-	0.4ms	233.6ms
Sentence-T5	-	1.2ms	223.5ms
ToCoTR	-	1.2ms	235.7ms

Table 6: Latency analysis for two spell-checker APIs and four retrieval systems. We retrieve the top 1000 results for MSMARCO dev queries with a single thread and then average over all the queries. The latency for two spell-checker API calls depends on server load and is difficult to precisely measure. For example, the latency of GG-spellchecker to call the API to return the full web page (not only the corrected queries) is marked $\star$.

B. Performance with Different Types of Simulated Typos

Typoed Queries	ToCoTR		coCondenser		SimLM	
	MRR@10	Recall@1000	MRR@10	Recall@1000	MRR@10	Recall@1000
w/o Typos	34.4	96.6	38.3	98.4	**41.1**	**98.7**
w/ Typos (Avg.)	**31.2**/-9.20%	**94.5**/-2.17%	22.4/-41.5%	84.9/-13.7%	23.9/-41.9%	84.2/-14.7%
RandInsert	**32.4**/-5.79%	**95.5**/-1.20%	22.9/-40.1%	85.5/-13.7%	23.8/-42.0%	84.2/-14.7%
RandDelete	**30.6**/-11.1%	**94.5**/-2.22%	22.7/-40.6%	86.2/-13.1%	24.7/-40.0%	86.0/-12.9%
RandSub	**30.1**/-12.3%	**93.6**/-3.17%	22.0/-40.6%	84.6/-12.4%	24.2/-41.1%	84.0/-14.9%
SwapNeighbor	**32.1**/-6.69%	**95.1**/-1.57%	21.6/-42.6%	83.5/-14.0%	22.9/-44.4%	82.5/-16.4%
SwapAdjacent	**30.9**/-10.1%	**94.0**/-2.68%	22.8/-43.7%	84.5/-15.1%	23.9/-42.0%	84.2/-14.7%

Table 7: Retrieval results on MSMARCO dev queries among ToCoTR, coCondenser, and SimLM concerning different types of simulated typos. Akin to Zhuang and Zuccon (2021), the drop rate (loss ratio in performance) on each misspelled query set to the original query set (denoted as w/o Typos) is also reported.