# Robustness of Neural Rankers to Typos: A Comparative Study

Shengyao Zhuang
The University of Queensland
Brisbane, QLD, Australia
s.zhuang@uq.edu.au

Xinyu Mao
The University of Queensland
Brisbane, QLD, Australia
xinyu.mao@uq.edu.au

Guido Zuccon
The University of Queensland
Brisbane, QLD, Australia
g.zuccon@uq.edu.au

## ABSTRACT

Recent advances in passage retrieval have seen the introduction of pre-trained language models (PLMs) based neural rankers. While generally very effective, little attention has been paid to the robustness of these rankers. In this paper, we study the effectiveness of state-of-the-art PLM rankers in presence of typos in queries, as an indication of the rankers' robustness. As of PLM rankers, we consider the two most promising directions explored in previous work: dense retrievers vs. sparse retrievers. We find that both types of rankers are very sensitive to queries with typos. We then apply an existing augmentation-based typos-aware training technique with the aim of creating typo-robust dense and sparse retrievers. We find that this simple technique only works for dense retrievers, while it hurts effectiveness when used on sparse retrievers.

## CCS CONCEPTS

• **Information systems → Evaluation of retrieval results**.

## KEYWORDS

dense retriever, sparse retriever, robustness on typo queries

**ACM Reference Format:**
Shengyao Zhuang, Xinyu Mao, and Guido Zuccon. 2022. Robustness of Neural Rankers to Typos: A Comparative Study. In *Australasian Document Computing Symposium (ADCS '22), December 15–16, 2022, Adelaide, SA, Australia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3572960.3572981

## 1 INTRODUCTION

Recent advances in neural information retrieval have seen the introduction of two divergent paradigms [16]: dense retrieval [8–10, 14, 15, 22, 24, 25, 31–33] and sparse retrieval [1, 4–7, 16, 19, 21, 35]. Dense retrieval refers to the fine-tuning of pre-trained language models (PLMs) which used to encode text into a dense vector representation. These dense vectors are then used for retrieval, by computing the similarity between a query and the passage encodings. Sparse retrieval refers to the reliance on (traditional) sparse inverted indexes to perform the retrieval. PLMs are used in the indexing phase, typically for expanding a document, i.e. adding additional semantically related unique terms not presented in a document, or for computing new weights for terms, e.g., increasing the term frequency of a term in a document.

While generally very effective, it is unclear if these neural methods are robust. For example, Sciavolino et al. [28] investigated the effectiveness of dense retrievers on queries that contain named entities and showed the methods have a hard time answering these sort of queries. Wu et al. [30] have examined the robustness of neural rankers to out-of-distribution data and adversarial attacks, showing these approaches are less robust than traditional probabilistic ranking models and learning-to-rank methods. Relevant to our paper is the work by Zhuang and Zuccon [34], who investigated the effectiveness of dense retrievers on queries that contain typos. Typos affect as high as 26% of queries in modern web search engines [29]. Zhuang and Zuccon [34] found that the losses in effectiveness caused by typos in queries are significant for dense retrievers. To remedy this, they proposed a method based on data augmentation for dense retrievers, called typos-aware training, which aimed at reducing the gap in effectiveness observed in dense retrievers when dealing with queries with typos, as opposed to queries that do not contain typos. Empirically, they demonstrated this new training method made the dense retrievers more robust to typos in queries. Importantly, however, their study was limited to dense retrievers trained with a standard BM25 hard negative sampling approach, thus it is unclear if this simple approach can be adapted to more advanced dense retriever training algorithms, or to sparse retrievers.

In this paper, we investigate whether sparse retrievers also present significant effectiveness gaps when faced with queries with typos, and whether these gaps are higher or lower than in dense retrievers. In addition, we also test whether the typos-aware training method, first proposed for and evaluated on dense retrievers, also generalises to sparse retrievers and more recent state-of-the-art dense retrievers that use more complex hard negative sampling strategies and pre-training. For this, we perform experiments considering a wide array of typos that could occur in English queries and we compare dense retrievers and sparse retrievers.

Code for reproducing the results presented in this paper can be found at https://github.com/ielab/typo-comparative-study

## 2 METHODOLOGY

### 2.1 Research Questions

There is no large-scale dataset that has enough queries with typos and associated labelled passages to allow us to train and evaluate retrieval models. Hence, we adopt the synthetic typo query generation framework by Zhuang and Zuccon [34]. We briefly detail these methods this method in Sections 2.2 and 2.3.

We design our experiments to answer the research questions:

- **RQ1**: Are current state-of-the-art sparse and dense retrievers robust to queries with typos? How do sparse retrievers compare in this respect to dense retrievers?

- **RQ2**: Do the improvements provided by typos-aware training on dense retrievers for queries with typos generalise to sparse retrievers and recent, advanced dense retrievers? Do models obtained with typos-aware training still retain their effectiveness on queries without typos?

To answer RQ1, we investigate the robustness of a wide array of dense and sparse retrievers. For the dense retrievers, we consider ANCE, DistilBERT-TASB, TCT-ColBERTv2, CoCondenser. For the sparse retrievers, we consider BM25, docTquery-T5, DeepImpact, uniCOIL, TILDEv2, SPLADEv2. We provide a description of each dense and sparse retriever in Sections 2.4 and 2.5.

To answer RQ2, we apply the typos-aware training strategy [34] to dense and sparse retrievers. Because of the requirement to re-train the sparse models from scratch for fair comparison, we only test typos-aware training on TILDEv2, uniCOIL, SPLADEv2 and CoCondenser[1] and then compare the effectiveness of these models trained with and without typos-aware on queries with and without typos. While this is a limited set of models, they are advance, highly performing models. To ensure fairness in comparison, we need that the training configurations (batch size, learning rate, random hard negatives, etc.,) are exactly the same for both models with and without typos-aware training. For this, we re-train all the models with an open-sourced retriever training tollkit Tevatron[2] [11]. We then implement typos-aware training based on the original training code by Zhuang and Zuccon [34]. For spares models, we use the same training configuration suggested in the toolkit examples which are provided by the original authors, and fine-tune a Huggingface BERT-base-uncased checkpoint from scratch. For CoCondenser, we use the CoCondenser checkpoint pre-trained on MS MARCO that is publicly available on Huggingface model hub[3] and we only conduct the two stages of fine-tuning. Due to limitations with the GPU infrastructure at our institution, we use a batch size of 32 instead of 64 as it was used in the original paper. For typos-aware training, we set the typo rate for the training queries to 0.5 and the types of typos are randomly sampled for each training query, as per original configurations [34].

## 2.2 Generating Queries with Typos

We follow Zhuang and Zuccon [34] in generating queries with typos for evaluating dense and sparse retrievers. For this, we create a query dataset from three existing datasets: MS MARCO passage ranking dev query set [20], the TREC Deep Learning Track Passage Retrieval Task 2019 [3] (DL 2019) and DL 2020 [2]. The original queries in these datasets do not contain typos[4]. From these, we generate queries with typos (separately, for each dataset) using five synthetic but realistic typo-generators: Random character Insertion (RandInsert), Random character deletion (RandDelete), Random character substitution (RandSub), Swap neighbour character (Swap-Neighbour), Swap adjacent keyboard character (SwapAdjacent). These typo-generators have been designed to imitate real world typos made by users on English queries submitted to a search engine [12]. Specifically, for each query in each dataset, we generate a typo by randomly sampling a typo-generator and also randomly sampling a word from the query to apply the typo-generator. By doing this, for each dataset we can obtain a version of the query set that contains typos. These new sets of queries with typos share the relevant passages with the corresponding queries without typos.

## 2.3 Typos-aware Training

For training typo-robust sparse and dense retrievers, we apply the data augmentation-based typos-aware training proposed by Zhuang and Zuccon [34]. For this, we maintain the key original training settings of each model unchanged and only apply the data augmentation from the typos-aware training to the queries used during training. For this, we use a typo-rate hyperparameter during the training phase which controls the amount of training queries into which a typo is injected. Following Zhuang and Zuccon [34], we set the typo rate to 0.5, i.e. each training query has a 50% chance to have a typo being injected. Injected typos are uniformly sampled from the typo generator. Therefore, both queries with and without typos are used in the model training phase: the goal is to minimize the training loss so that the model will be robust to different types of typos. For more details about the typos-aware training, we refer the readers to the corresponding original paper [34].

## 2.4 Considered Dense Retrievers

- `ANCE` [31]: ANCE is a RoBERTa-based [18] dense retriever trained with dynamic dense index refreshing and hard negative sampling. ANCE has been widely used as a comparative baseline.
- `DistilBERT-TASB` [13]: A DistilBERT-based [27][5] dense retriever trained with ranking knowledge distillation from a large cross-encoder teacher model, using a bi-encoder dense retriever student model.
- `TCT-ColBERTv2` [17]: A BERT-based dense retriever trained with ranking knowledge distillation from both a cross-encoder teacher model and a ColBERT [15] (a strong multi-vector dense retriever) teacher model.
- `CoCondenser` [9]: State-of-the-art dense retriever trained with two stages of pre-training and two stages of fine-tuning. For the pre-training phase, the first stage is the Condenser training [8] where a CLS token enhanced BERT is pre-trained on the masked language modelling task. The second stage is the corpus aware pre-training where contrastive loss is used so that the representations of two sentences from the same passage are trained to be close to each other. After the pre-training, the two stages of fine-tuning are conducted on the target task with labelled data. In the first fine-tuning stage, a strong dense retriever is obtained for sampling hard negative training instances. Then, in the second stage, the final dense retriever model is trained with hard negatives from the first stage.

## 2.5 Considered Sparse Retrievers

- `BM25` [26]: A classic bag-of-words zero shot sparse retrieval model.
- `docTquery-T5` [21]: a T5 language model [23] fine-tuned to generate relevant queries from the given passage. Specifically, for each passage, 40 queries are generated and appended to the passage. Then BM25 is used on the extended passage collection.

[1]We note that typos-aware training was evaluated for other dense retrievers in previous work [34].
[2]https://github.com/texttron/tevatron/tree/main/examples/coCondenser-marco
[3]Huggingface hub string: Luyu/co-condenser-marco.
[4]With the exception of a handful of queries in the MS MARCO dev set.
[5]A half-size BERT trained with knowledge distillation on the language modelling task.

| Model | | MS MARCO | TREC DL 2019 | | TREC DL 2020 | |
|---|---|---|---|---|---|---|
| | | MRR@10 | nDCG@10 | MAP | nDCG@10 | MAP |
| Sparse | BM25 | .187/.095 (-49.3%) | .497/.256 (-48.5%) | .290/.147 (-49.3%) | .487/.291 (-40.2%) | .287/.168 (-41.5%) |
| | docTquery-T5 | .282/.147 (-47.6%) | .634/.339 (-46.4%) | .405/.201 (-48.9%) | .624/.409 (-34.5%) | .417/.266 (-36.3%) |
| | DeepImpact | .325/.147 (-54.9%) | .694/.398 (-42.8%) | .457/.242 (-47.0%) | .650/.417 (-35.8%) | .426/.268 (-37.1%) |
| | uniCOIL | .351/.188 (-46.4%) | .703/.395 (-43.8%) | .462/.235 (-49.1%) | .674/.454 (-32.7%) | .444/.279 (-37.2%) |
| | TILDEv2 | .337/.181 (-46.2%) | .647/.371 (-42.6%) | .417/.219 (-47.4%) | .653/.437 (-33.1%) | .427/.262 (-38.7%) |
| | SPLADEv2 | .368/.210 (-43.1%) | .728/.464 (-36.3%) | .485/.285 (-41.1%) | .710/.503 (-29.2%) | .488/.322 (-34.0%) |
| Dense | ANCE | .330/.200 (-39.3%) | .645/.448 (-30.5%) | .371/.247 (-33.4%) | .641/.461 (-28.1%) | .403/.278 (-31.0%) |
| | DistilBERT-TASB | .344/.184 (-46.5%) | .721/.441 (-38.8%) | .459/.264 (-42.5%) | .683/.470 (-31.2%) | .468/.303 (-35.1%) |
| | TCT-ColBERTv2 | .358/.199 (-44.4%) | .720/.449 (-37.6%) | .447/.256 (-42.7%) | .689/.471 (-31.6%) | .475/.302 (-36.4%) |
| | CoCondenser | .383/.221 (-42.4%) | .715/.461 (-35.5%) | .453/.276 (-40.7%) | .679/.479 (-29.4%) | .479/.315 (-34.2%) |

**Table 1: The effectiveness of dense vs. sparse retrievers on queries with and without typos (results to be read as: <without/with>). The percentage of loss (drop rates) between queries without and with typos are reported in brackets. For every method, effectiveness on queries with typos is statistically significantly lower than for the queries without typos. Other statistically significant comparisons of interest are mentioned directly in the text write-up.**

- `DeepImpact` [19]: it estimates a contextualized impact score for each token in a passage using a fine-tuned BERT model. The impact scores are used to substitute the term-weighting scores in the inverted index so that sparse retrieval can be performed with the contextualized impact scores. The impact scores of query tokens are just one-hot vector given by BERT tokenization. DeepImpact also uses the docTquery-T5 expanded collection.
- `uniCOIL` [16]: uniCOIL learns impact scores for both query tokens and passage tokens using contrastive loss, unlike DeepImpact that only predicts impact scores for passage tokens but also uses learned BERT model to estimate impact scores of query tokens. uniCOIL also uses the docTquery-T5 expanded collection.
- `TILDEv2` [35]: TILDEv2 can be regarded as a combination of the key characteristics of DeepImpact and uniCOIL, where the query encoder is just the BERT tokenizer and the passage encoder is the BERT model fine-tuned with contrastive loss. We note that the original authors use TILDEv2 as a second stage re-ranker; we instead convert it into a first stage retriever by using the same retrieval pipeline as uniCOIL for fair comparison.
- `SPLADEv2` [6]: SPLADEv2 is the current state-of-the-art sparse retriever. It learns impact scores with contrastive loss and knowledge distillation from a cross-encoder teacher. Unlike DeepImpact, TILDEv2 and uniCOIL, it does not rely on docTquery-T5 to expand passages with more tokens but it directly estimates impact scores for each passage over the entire BERT vocabulary.

## 2.6 Dataset and evaluation

All models are trained with the MS MARCO passage training data which consists of around 500k training queries. We evaluate all models using the 6,980 MS MARCO dev queries, the 43 TREC 2019 Deep Learning Track queries (TREC DL 2019) [3] and the 54 TREC 2020 Deep Learning Track (TREC DL 2020) queries [2]. For the evaluation of queries with typos, we repeat the synthetic typo-generation process 10 times for each dataset and report the average scores. Following official guidelines for the datasets, we use MRR@10 to evaluate MS MARCO queries, and nDCG@10 and MAP for TREC DL 2019 and TREC DL 2020 queries. Statistically significant differences between methods' results are detected using a two-tailed paired t-test with $p < 0.05$ and Bonferroni correction.

## 3 RESULTS

Table 1 reports the results for **RQ1**. The effectiveness of all methods sharply decreases once typos are introduced in the queries: no matter whether dense or sparse retrievers are used, drops in effectiveness are substantial and always statistically significant. Dense retrievers however do appear to have relative losses that are more modest than those of sparse retrievers. Our hypothesis is that sparse retrievers exhibit higher relative losses on queries with typos than dense retrievers because sparse retrievers rely on exact term matching between query and passage tokens, and a typo introduces a lexical mismatch (vocabulary mismatch).

We also note that while on queries without typos differences between certain methods are remarkable, on queries with typos differences disappear. For example, DeepImpact is more effective than docTquery-T5 in general (see MRR@10 for MS MARCO results, for which differences are significant); however when queries with typos are considered, DeepImpact and docTquery-T5 achieve the same effectiveness (no statistically significant difference). This finding is even more consistent for dense retrievers. While in general there are significant differences between the effectiveness of these methods, when queries with typos are considered, the only significant differences are found for the CoCondenser vs. the other dense retrievers on the MS MARCO dev queries.

Interestingly, although ANCE has the lowest effectiveness amongst the considered dense retrievers on queries without typos, it exhibits the highest robustness to typos as it is consistently characterised by the lowest effectiveness drop across all datasets and metrics: at times, it even outperforms stronger dense retrievers on queries with typos.

Hence, in answer to **RQ1**, we conclude that current state-of-the-art sparse and dense retrievers are not robust to queries with typos. Dense retrievers exhibit slightly higher robustness than sparse

| Model | MS MARCO | TREC DL 2019 | | TREC DL 2020 | |
|---|---|---|---|---|---|
| | MRR@10 | nDCG@10 | MAP | nDCG@10 | MAP |
| TILDEv2 | **.337**/.181 (-46.2%) | .647/.371 (-42.6%) | .417/.219 (-47.4%) | .653/.437 (-33.1%) | .427/.262 (-38.7%) |
| TILDEv2 (typos-aware) | .327/**.189** (-42.2%) | .652/.388(-40.5%) | .399/.226 (-43.3%) | .622/.442 (-29.0%) | .396/.257 (-35.1%) |
| uniCOIL | **.351**/.190 (-46.0%) | **.700**/.401 (-42.6%) | .457/.240 (-47.4%) | .688/.449 (-34.7%) | .456/.278 (-39.1%) |
| uniCOIL (typos-aware) | .343/**.208** (-39.4%) | .659/.439 (-33.4%) | .427/.266 (-37.6%) | .659/.479 (-27.4%) | .425/.286 (-32.9%) |
| SPLADEv2 | .355/.180 (-49.3%) | .670/.362 (-46.0%) | .434/.223 (-48.6%) | .624/.390 (-37.5%) | .421/.251 (-40.4%) |
| SPLADEv2 (typos-aware) | .356/**.237** (-33.4%) | .690/**.482**(-30.1%) | .433/**.291** (-32.8%) | .623/.459 (-26.3%) | .408/.291 (-28.7%) |
| CoCondenser | .376/.193(-48.6%) | .689/.402 (-41.7%) | .447/.244 (-45.4%) | .675/.436 (-35.4%) | .476/.287 (-39.8%) |
| CoCondenser (typos-aware) | .376/**.265** (-29.3%) | .703/**.507** (-27.9%) | .450/**.313** (-30.4%) | .656/.512 (-22.0%) | .457/.338 (-26.0%) |

**Table 2: The effectiveness of sparse retrievers (TILDEv2, uniCOIL, SPLADEv2) and a dense retriever (CoCondenser) with and without typos-aware training. (We re-trained the original models for fair comparison). Effectiveness is reported on queries with and without typos (results to be read as: <without/with>). For each method, statistically significant differences between our trained and our typos-aware trained models are indicated in bold.**

retrievers, and among them, ANCE shows the least degradation due to typos.

Table 2 reports the results for **RQ2**, where we compare sparse retrievers (TILDEv2, uniCOIL and SPLADEv2) and the state-of-the-art dense retriever (CoCondenser) trained in their standard fashion vs. with typos-aware training. Recall that for this comparison we could not use the checkpoints made available by the original authors, due to differences in training parameters with those we could use for typos-aware training – and thus resulted in retraining the basic models.

We start the analysis by comparing the results obtained by the original model checkpoints (Table 1) and those obtained when we trained these models (Table 2, our trained). For our trained version of TILDEv2 and uniCOIL we obtained results that are substantially similar to those of the original checkpoint, with no difference being statistically significant. However, for SPLADEv2 and CoCondenser, our trained model exhibits a lower effectiveness than the original checkpoint. We consider this is due to the smaller batch size we used for training: batch size is important for the contrastive loss with in-batch negative samples [22], used by SPLADEv2 and CoCondenser. Nevertheless, the training with and without typos-aware for SPLADEv2 and CoCondenser are conducted under the exactly same conditions and hyperparameters. We believe this controlled experimental setting is valid for studying the impact of typos-aware training, and thus we proceed with using our trained SPLADEv2 and CoCondenser checkpoint also in the analysis that follows for RQ2.

Next, we compare the results of the original training of sparse and dense retrievers with their typos-aware training. Typos-aware training is less effective for TILDEv2 and uniCOIL. The improvement for queries with typos is marginal. When comparing to the original training, TILDEv2 and uniCOIL trained with typos-aware training achieve lower effectiveness, although the only statistically significant difference is found for MRR@10 on MS MARCO (for both TILDEv2 and uniCOIL) and nDCG@10 on TREC DL2019 (for uniCOIL). Hence, we conclude that typos-aware training has little effect to TILDEv2 and uniCOIL.

In contrast, typos-aware training appears to work well for SPLADEv2 and CoCondenser. When used on queries without typos, both the SPLADEv2 and CoCondenser trained with typos-aware training exhibit no statistically significant differences with the models we trained with standard training. On the other hand, on queries with typos, SPLADEv2 and CoCondenser trained with typos-aware training exhibit much higher effectiveness than their counterpart with standard training, and these differences are statistically significant on MS MARCO and TREC DL 2019.

It is notable that, although TILDEv2, uniCOIL, and SPLADEv2 are all sparse retrievers, we obtained considerably different results when comparing TILDEv2 and uniCOIL with SPLADEv2. The typos-aware training is more effective for SPLADEv2 than that for TILDEv2 and uniCOIL. We argue that this is due to the fact that TILDEv2 and uniCOIL only have a passage expansion step as data preprocessing strategy before training. On the other hand, SPLADEv2 learns how to expand queries and passages on-the-fly during training. Hence, SPLADEv2 is able to learn a query and passage expansion pattern that tolerates the typos in queries where TILDEv2 and uniCOIL cannot. This hypothesis is clarified by the following examples. The test set contains the query with typo *"who kliled nicholas ii of russia"* [6]: here, the term *"kliled"* is a typo of the term *"killed"*. In this case, SPLADEv2 trained with typos-aware training expands this query with the correctly spelled term *"killed"* along with other synonyms such as *"murder"*. However, our trained uniCOIL and TILDEv2, no matter whether they are trained with or without typos-aware training, cannot match the correctly spelled term *"killed"* in a passage with the terms in the misspelled query since they cannot expand the query with correctly spelled terms. On the other hand, SPLADEv2 without typos-aware training also does not provide an expansion of this query that contains the correctly spelled terms. It is then logical that the improvements provided by typos-aware training on SPLADEv2 are significant when in presence of queries that contain typos.

Hence, in answer to **RQ2**, we found that the augmentation-based typos-aware training [34] is effective to improve the robustness of the current state-of-the-art sparse and dense retrievers (SPLADEv2 and CoCondenser) to queries with typos. On the other hand, we found that this training regime hinders the effectiveness of the considered sparse retrievers that only perform passage expansion

[6]The original query is in the DL2020 dataset, query id 1043135.

(TILDEv2 and uniCOIL). We believe this finding would likely also be applicable to the other sparse retrievers we did not experiment with (e.g., DeepImpact), which before training also use passage expansion only. Our results instead suggest that the end-to-end combined passage and query expansion training strategy used by SPLADEv2 is beneficial when coupled with the typos-aware training strategy.

Our work did not examine the effectiveness of typos-aware training on more retrievers due to time limitation, and measuring the effect of different types of typos on the training process was also not considered. As shown by our experiments, the augmentation-based typos-aware training paradigm is detrimental to TILDEv2 and uniCOIL which are representative neural sparse retrievers. There were a number of limitations related to the typos we considered. First, as no large scale dataset that has enough queries with typos and associated labelled passages was available, we had to resort to synthetically generate typos. While these typos were realistic, they only considered errors made with respect to insertion/deletion of characters, and substitution of characters with those adjacent on the keyboard; however, no phonetically compatible misspellings[7] were considered. We also only examined the possibility that a query contained a single typo, while typos could be multiple in real queries. Another limitation is that we considered typos that can occur on languages based on western characters. We note that typos in other languages, e.g. Chinese, would follow different patterns.

## 4 CONCLUSIONS

In this paper, we contributed a comparative study of the effectiveness of sparse and dense retrievers on queries with typos. With this respect, we found that both sparse and dense retrievers are not robust to typos in queries, and sparse retrievers are generally more sensitive than dense retrievers.

We also evaluated a data augmentation-based typos-aware training method [34], which was previously shown effective in making dense retrievers more robust to queries with typos. We applied this method to the sparse retrievers TILDEv2, uniCOIL, SPLADEv2 and the state-of-the-art dense retriever CoCondenser. We found that this training method is effective to make the CoCondenser more robust to typos in queries: This is a novel contribution because the CoCondenser uses a more sophisticated training strategy than the dense retrievers for which typos-aware training was evaluated, and thus it was unclear whether improvements would still apply. On the other hand, however, when applied to the sparse retrievers TILDEv2 and uniCOIL, we found that typos-aware training is less effective for queries with typos, and it slightly hurts their effectiveness on queries that do not contain typos. Interestingly, among the considered sparse models, SPLADEv2 is the only sparse retriever to be effective when typos-aware training is used. We then provided an hypothesis that explains why this is so, along with an example to support our intuition. Key to SPLADEv2 effectiveness when combined with typos-aware training is SPLADEv2's strategy for both query and passage expansion. These findings more generally suggest that typos-aware training could also be effective for sparse retrievers if the end-to-end query and passage expansion strategy is used during training.

[7] i.e., one that can be pronounced in the same way as the original word, e.g. *hurd* and *herd*.

*Acknowledgements.* Shengyao Zhuang is supported by a UQ Earmarked PhD Scholarship. Xinyu Mao is supported by the Australian Research Council Discovery Project DP210104043.

## REFERENCES

[1] Yang Bai, Xiaoguang Li, Gang Wang, Chaoliang Zhang, Lifeng Shang, Jun Xu, Zhaowei Wang, Fangshan Wang, and Qun Liu. 2020. SparTerm: Learning term-based sparse representation for fast text retrieval. *arXiv preprint arXiv:2010.00768* (2020).

[2] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2020. Overview of the TREC 2020 Deep Learning Track. In *Proceedings of the Twenty-Ninth Text REtrieval Conference (NIST Special Publication)*. National Institute of Standards and Technology (NIST).

[3] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the TREC 2019 Deep Learning Track. In *Proceedings of the Twenty-Ninth Text REtrieval Conference (NIST Special Publication)*. National Institute of Standards and Technology (NIST).

[4] Zhuyun Dai and Jamie Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint arXiv:1910.10687* (2019).

[5] Zhuyun Dai and Jamie Callan. 2020. Context-aware document term weighting for ad-hoc search. In *Proceedings of The Web Conference 2020*. 1897–1907.

[6] Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086* (2021).

[7] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2288–2292.

[8] Luyu Gao and Jamie Callan. 2021. Condenser: a Pre-training Architecture for Dense Retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 981–993.

[9] Luyu Gao and Jamie Callan. 2021. Unsupervised corpus aware language model pre-training for dense passage retrieval. *arXiv preprint arXiv:2108.05540* (2021).

[10] Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2021. Complementing Lexical Retrieval with Semantic Residual Embedding. In *43rd European Conference on IR Research, ECIR 2021*.

[11] Luyu Gao, Xueguang Ma, Jimmy J. Lin, and Jamie Callan. 2022. Tevatron: An Efficient and Flexible Toolkit for Dense Retrieval. *ArXiv* abs/2203.05765 (2022).

[12] 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*. 1261–1264.

[13] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving efficient neural ranking models with cross-architecture knowledge distillation. *arXiv preprint arXiv:2010.02666* (2020).

[14] 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)*. Association for Computational Linguistics, Online, 6769–6781. https://doi.org/10.18653/v1/2020.emnlp-main.550

[15] 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*. 39–48.

[16] Jimmy Lin and Xueguang Ma. 2021. A few brief notes on deepimpact, coil, and a conceptual framework for information retrieval techniques. *arXiv preprint arXiv:2106.14807* (2021).

[17] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-Batch Negatives for Knowledge Distillation with Tightly-Coupled Teachers for Dense Retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*. Association for Computational Linguistics, Online, 163–173. https://doi.org/10.18653/v1/2021.repl4nlp-1.17

[18] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv* abs/1907.11692 (2019).

[19] A Mallia, O Khattab, T Suel, and N Tonellotto. 2021. Learning Passage Impacts for Inverted Indexes. In *44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2021*. Association for Computing Machinery, Inc, 1723–1727.

[20] 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 *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (CEUR Workshop Proceedings, Vol. 1773)*. CEUR-WS.org.

[21] Rodrigo Nogueira and Jimmy Lin. 2019. From doc2query to docTTTTTquery.

[22] 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*. 5835–5847.

[23] Colin Raffel, Noam M. 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. *ArXiv* abs/1910.10683 (2020).

[24] Ruiyang Ren, Shangwen Lv, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. PAIR: Leveraging Passage-Centric Similarity Relation for Improving Dense Passage Retrieval. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. 2173–2183.

[25] 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*. 2825–2835.

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

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

[28] Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. 2021. Simple Entity-Centric Questions Challenge Dense Retrievers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 6138–6148.

[29] Peiling Wang, Michael W Berry, and Yiheng Yang. 2003. Mining longitudinal Web queries: Trends and patterns. *Journal of the american Society for Information Science and technology* 54, 8 (2003), 743–758.

[30] Chen Wu, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. 2021. Are Neural Ranking Models Robust? *arXiv preprint arXiv:2108.05018* (2021).

[31] 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*.

[32] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing Dense Retrieval Model Training with Hard Negatives. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2021).

[33] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. RepBERT: Contextualized text embeddings for first-stage retrieval. *arXiv preprint arXiv:2006.15498* (2020).

[34] 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*. 2836–2842.

[35] Shengyao Zhuang and Guido Zuccon. 2021. Fast passage re-ranking with contextualized exact term matching and efficient passage expansion. *arXiv preprint arXiv:2108.08513* (2021).