LREC-COLING 2024

MLPD 四川大學 SICHUAN UNIVERSITY

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

20-25 May, 2024

Contents

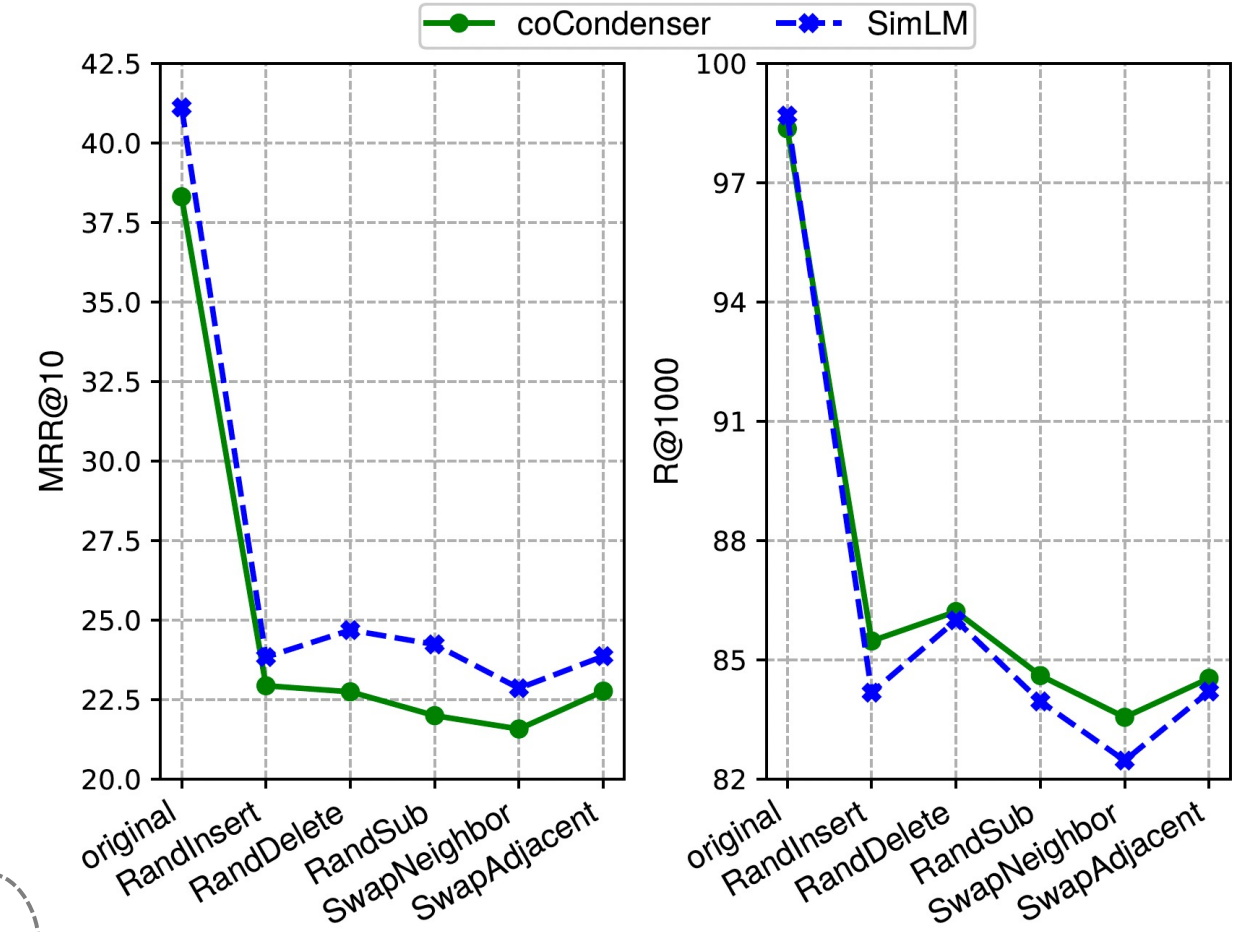
Background

- Misspelling Queries

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

- Ranking Performance in "Drop"

Typoed queries resulting from the users' mistyping words or phonetic typing errors exist widely in search behaviors.



MRR@10 and R@1000 results on MSMARCO.
A significant drop in effectiveness across different types of simulated typos on queries.

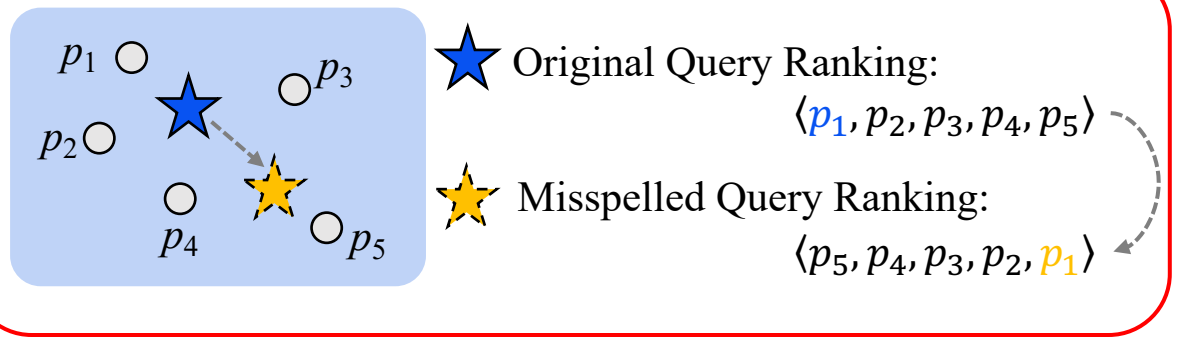
Background

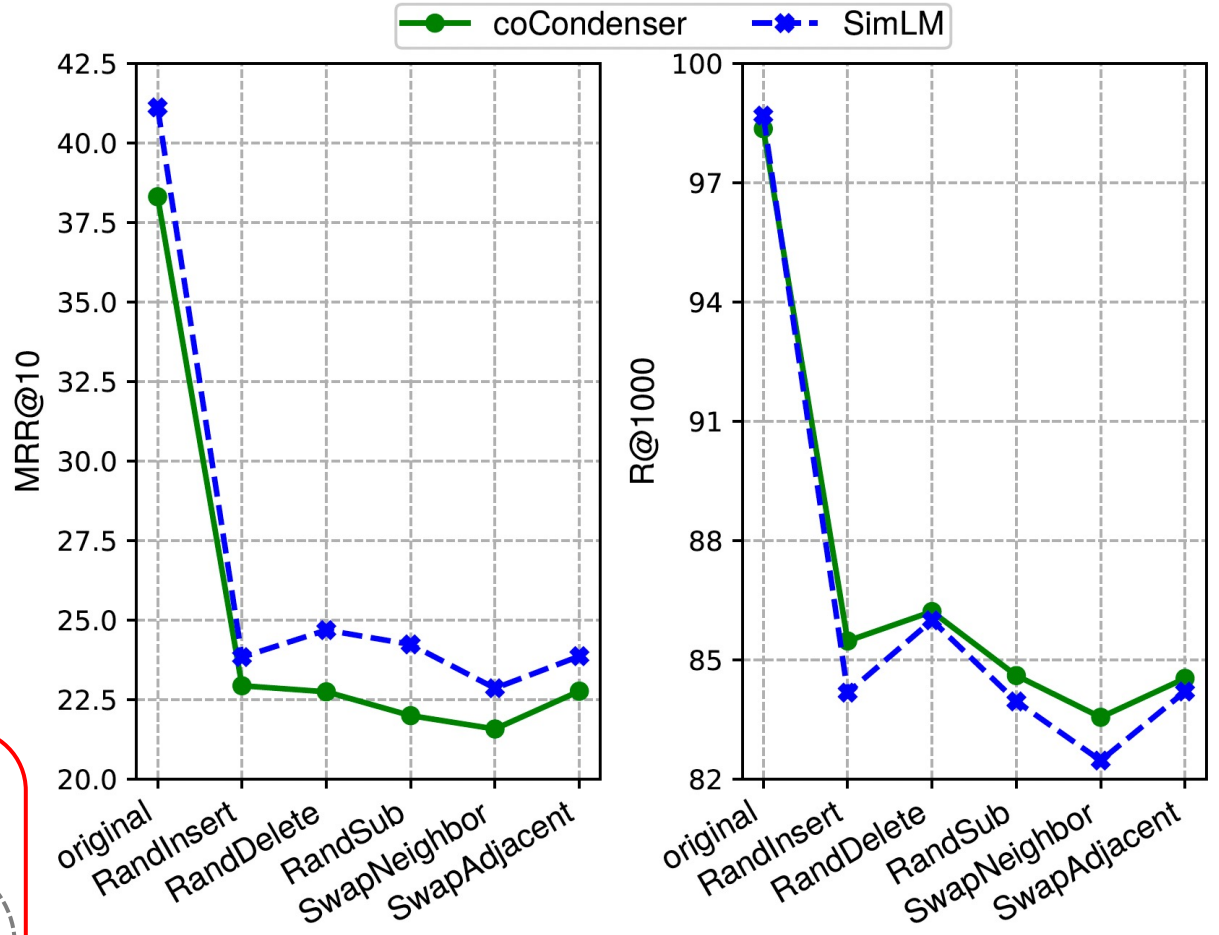
- Misspelling Queries

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

- Ranking Performance in "Drop"

Typoed queries resulting from the users' mistyping words or phonetic typing errors exist widely in search behaviors.

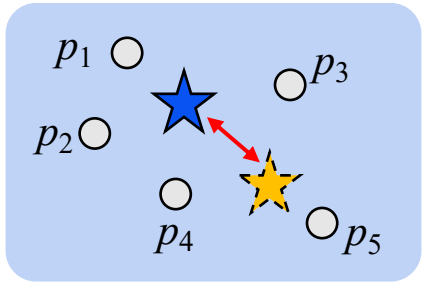




MRR@10 and R@1000 results on MSMARCO.
A significant drop in effectiveness across different types of simulated typos on queries.

Motivation

- Existing Methods



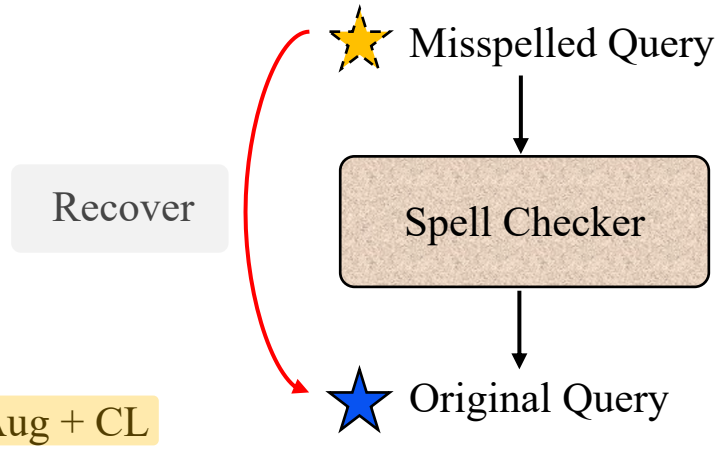
Original Query Ranking:
$\langle p_1, p_2, p_3, p_4, p_5 \rangle$

Misspelled Query Ranking:
$\langle p_5, p_4, p_3, p_2, p_1 \rangle$

Maximize agreement

- Data augmentation: BERT + Aug
- Contrastive learning to push a typoed query close to its original variation: BERT + Aug + CL
- Self-teaching with supervised label in KL-divergence: CharacterBERT + ST
- Local ranking alignment: RoDR

Industry soulution



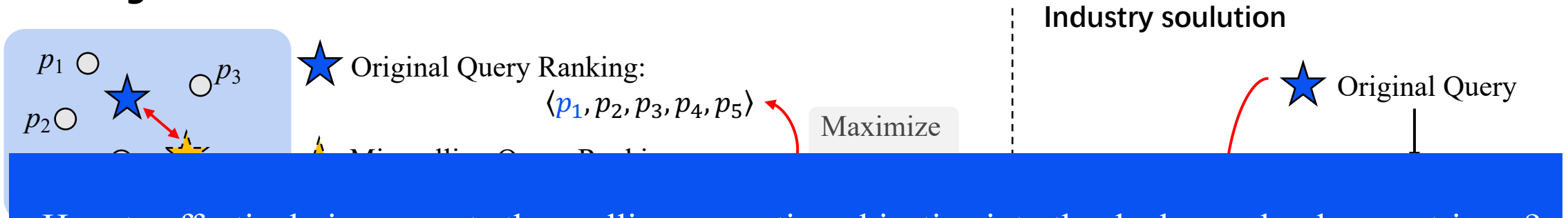
Observation 1: Simply aligning the latent embeddings or ranking differences between the original and misspelled queries is inadequate for sophisticated training retrievers such as coCondenser and SimLM.

Observation 2: The spell-checker and dense retriever are optimized as separate models. If the spell-checker is subpar, this will result in a decreased ranking performance.

Motivation

MLPD

- Existing Methods



How to effectively incorporate the spelling correction objective into the dual-encoder dense retriever?

- Base model selection?
- How to improve the effectiveness of the spell correction?
- Training strategy?

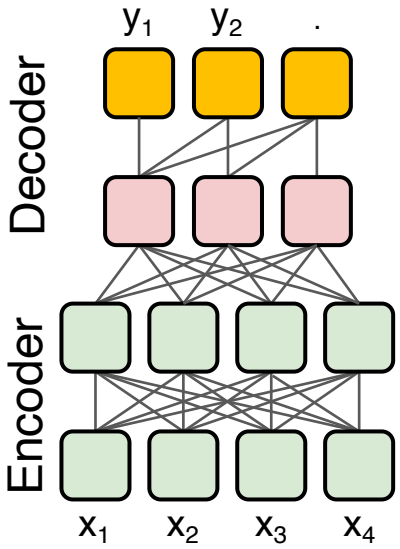
Observation 1: Simply aligning the latent embeddings or ranking differences between the original and misspelled queries is inadequate for sophisticated training retrievers such as coCondenser and SimLM.

Observation 2: The spell-checker and dense retriever are optimized as separate models. If the spell-checker is subpar, this will result in a decreased ranking performance.

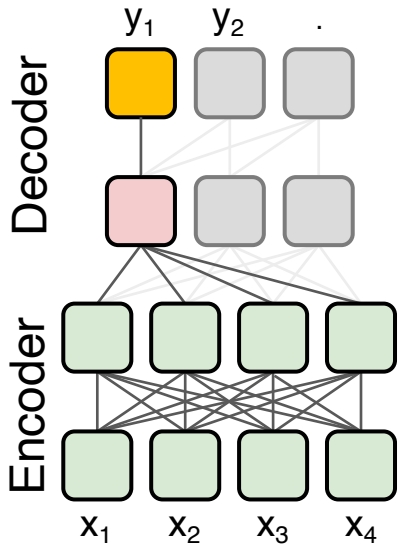
Our method: ToCoTR

- Base model selection

- Spelling correction is formulated as a monolingual translation task and treated with an encoder-decoder based model[1].
- The encoder-decoder sentence embedding model have proved to be a promising architecture[2].



T5 Encoder-Decoder
(Spell Correction)



T5 Encoder-Decoder First
(Sentence Embedding)

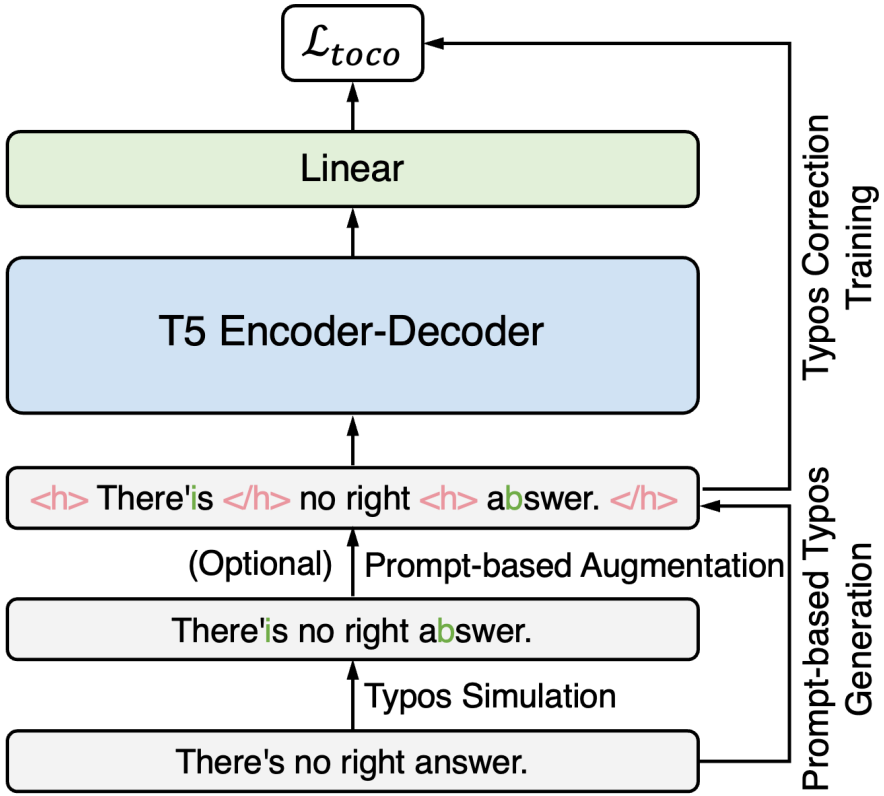
- The T5-style model does not place a special symbol (e.g., [CLS] in BERT) at the beginning of the text sequence.
- To obtain the decoder output, the input text is fed into the encoder, and the standard "start" symbol (first token) is fed as the first decoder input.

[1] Rothe S, Mallinson J, Malmi E, et al. A Simple Recipe for Multilingual Grammatical Error Correction. ACL 2021.

[2] Ni J, Abrego G H, Constant N, et al. Sentence-T5: Scalable Sentence Encoders from Pre-trained Text-to-Text Models. ACL (Findings) 2022.

Our method: ToCoTR

- Prompt-based typos correction training

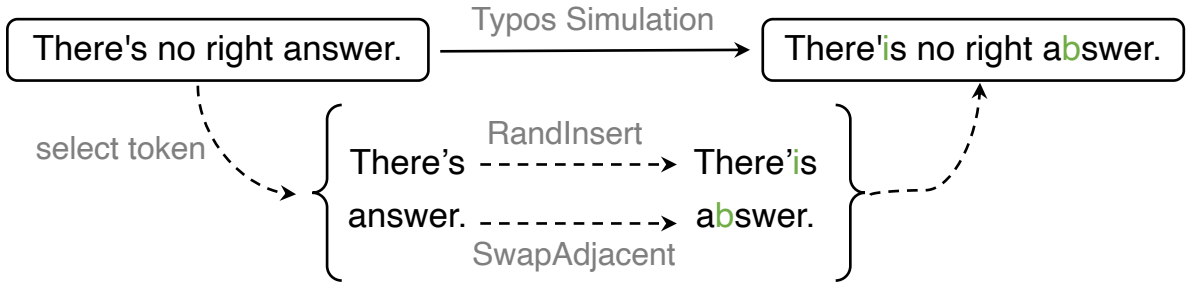


$$\mathcal{L}_{toco} = -\log \sum_{k=1}^{|C|} (\mathbf{P}(Y_k \mid X_k^{typo}; \theta))$$

S1: Given a set of context $C = \{X_1, X_2, \ldots, X_{|C|}\}$ and randomly sampling samples to conduct the prompt-based typos generation at a predetermined rate value (in our experiment, the rate equals 80%);

S2: For each selected source text $X_k = (x_1, x_2, \ldots, x_t)$, choose $\alpha|X_k|$ token positions at random to simulate typos;

S3: If the t-th token is chosen then use a randomly selected typos generator to inject the typos, including RandInsert, RandDelete, RandSub, SwapNeighbor, SwapAdjacent;



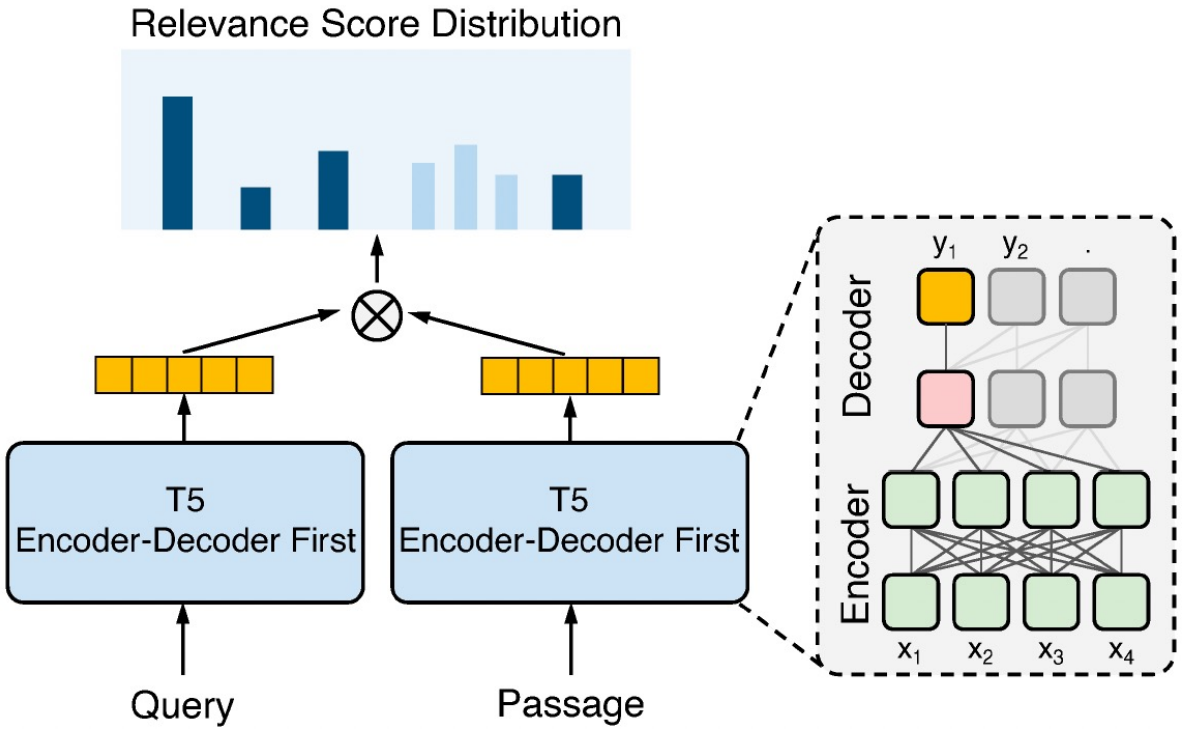
S4: Add special symbols to the selected typo token according to the augmentation template "<h> [X] </h>" to highlight the errors;

There'is ⟶ <h> There'is </h> ⋮ abswer. ⟶ <h> abswer. </h>

S5: Training with $\mathcal{L}_{toco}$

Our method: ToCoTR

- Dual-encoder architecture



- Given a query q_i, passage retrieval aims to return a sorted list of the n most relevant passages $L = [p_1, p_2, \dots, p_n]$ from a large set $D = \{p_i\}_{i=1}^{m}$ according to the relevance score of the retrieval model.

- For dense retriever training, we assume a set of binary positive correlation judgments as supervised signals, denoted by

$$R = \langle q_i, p_i^+, \{p_{i,1}^-, p_{i,2}^-, \dots, p_{i,s}^-\} \rangle$$

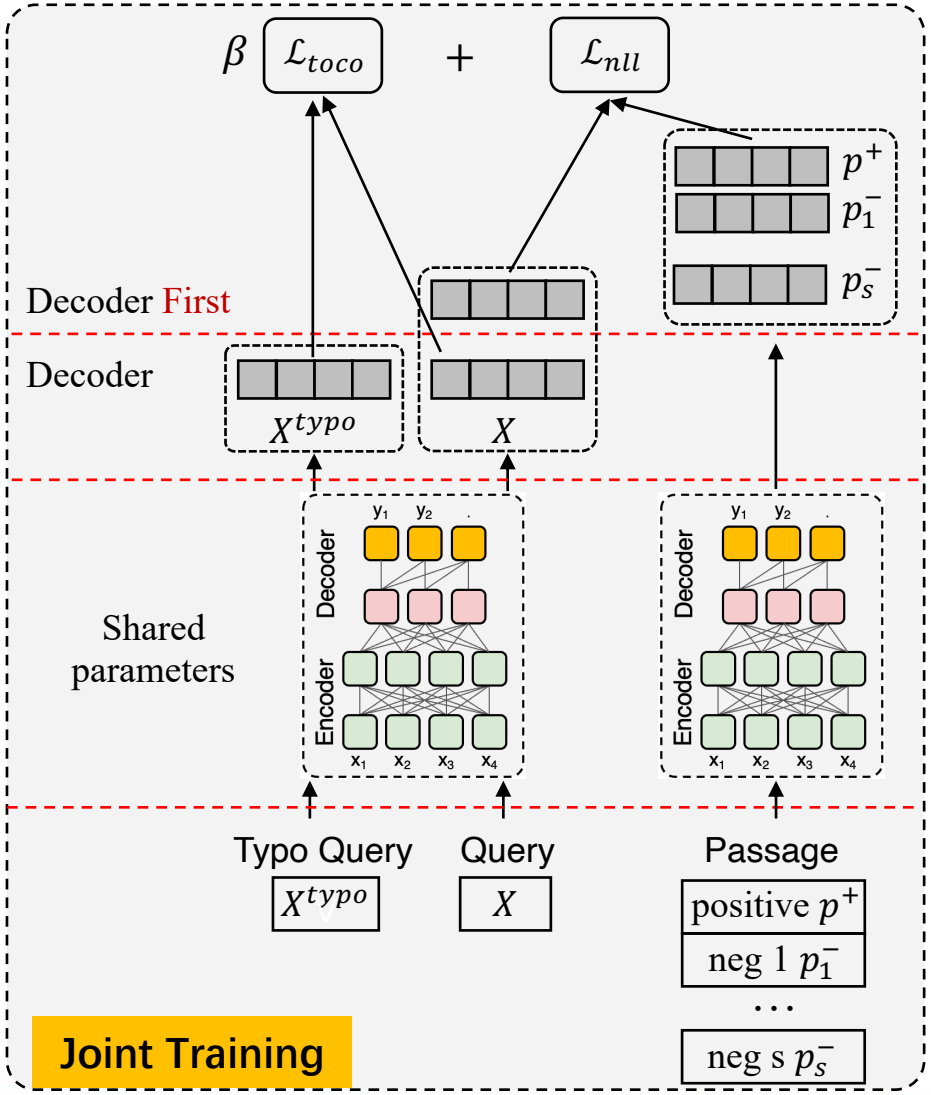
where p_i^+ denotes the relevant passages and p_i^- denotes the irrelevant passages for query q_i.

- To optimize the dense retriever, the negative log-likelihood (NLL) loss is applied:

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

Our method: ToCoTR

- Training strategy

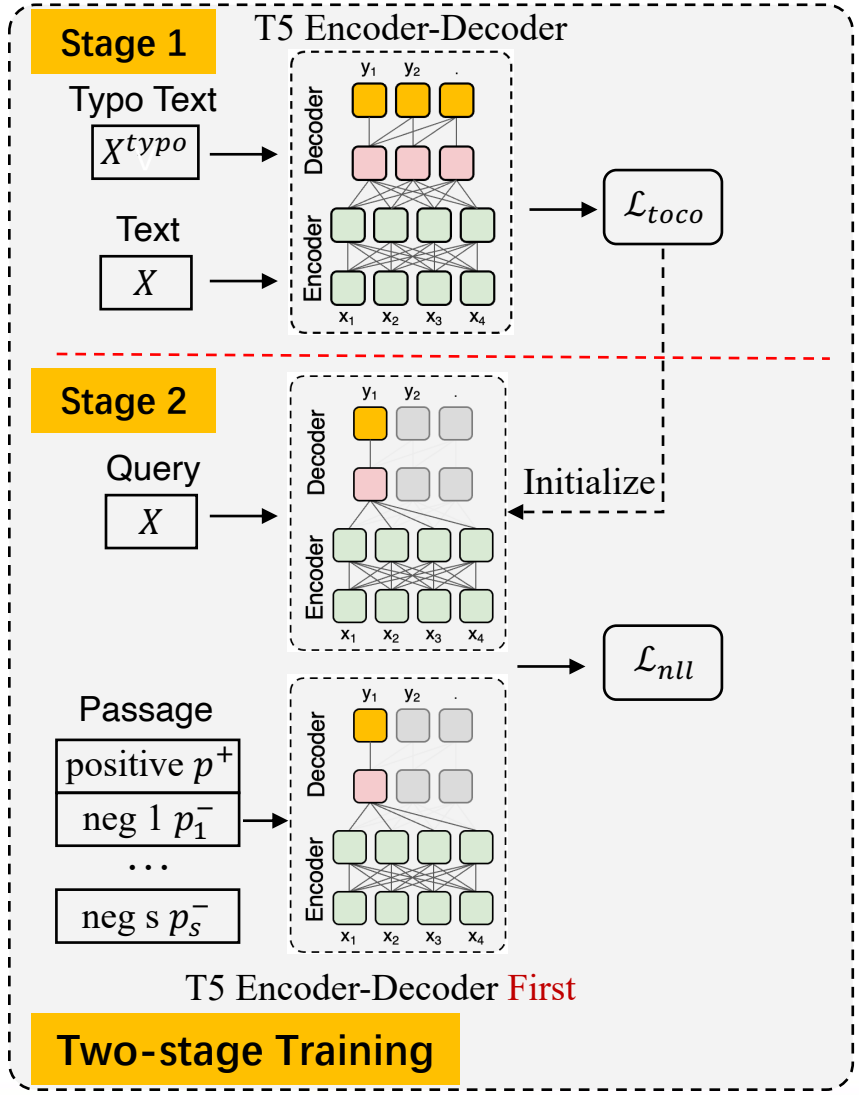


Two different training strategies:

- Joint training (Left)
- Two-stage training (Right)

Each training strategies can be coupled with existing training techniques:

- Self-teaching
- Hard-Negative mining



Results & Analysis

- Datasets

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

- MSMARCO
  Source: queries sampled from Bing search logs and annotated with binary relevant passages
  Typo Query: randomly generated 10 sets by repeating the typos simulation

- TREC 2019
  Source: queries sampled from Bing search logs and annotated with four-level relevant annotations (Same corpus with MSMARCO)
  Typo Query: randomly generated 10 sets by repeating the typos simulation

- ANTIQUE (zero-shot validate)
  Source: non-factual questions and answers from a community answering service, where questions and answers are manual four-level relevance annotations
  Typo Query: sampling three misspelling variations from manually validated typoed questions by released researchers, including SwapNeighbor, SwapAdjacent, RandSub

Results & Analysis

- Comparison with the retrievers

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**‡

‡ indicate significant differences with the second-best score (underlined) at p-value < 0.05

Results & Analysis

- Comparison with the retrievers involving spell-checkers

Methods	w/o typos		w/ typos	
	MRR@10	R@1000	MRR@10	R@1000
CharacterBERT	32.1	94.8	16.0	72.2
pyspellchecker → CharacterBERT	27.3	88.5	23.0	81.9
MS-Spellchecker → CharacterBERT	32.0	94.6	29.9	91.3
GG-Spellchecker → CharacterBERT	<u>32.2</u>	94.8	29.4	90.4
Sentence-T5	32.1	95.9	18.5	80.6
pyspellchecker → Sentence-T5	28.4	91.9	24.7	87.0
MS-Spellchecker → Sentence-T5	32.0	<u>95.9</u>	<u>30.5</u>	<u>93.6</u>
GG-Spellchecker → Sentence-T5	32.0	<u>95.9</u>	29.9	93.1
ToCoTR (two-stage)	**34.4**‡	**96.6**‡	**31.3**	**94.6**‡

‡ indicate significant differences with the second-best score (underlined) at p-value < 0.05

- pyspellchecker: a rule-based spell-checking toolkit that relies on dictionary-based rule sets;
- MS-Spellchecker: Microsoft Bing Spell Check API, it utilizes machine learning and statistical machine translation to provide corrections;
- GG-Spellchecker: Google Search API, it has been shown in previous research to be possibly the most useful spell corrections[1];

[1] Hagen M, Potthast M, Gohsen M, et al. A large-scale query spelling correction corpus. ACM SIGIR 2017.

Results & Analysis

- Ablation study

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↓	95.7↓	29.8↓	93.2↓
w/o Self-Teaching	34.1	97.0	27.3↓	91.9↓
w/o Prompt	34.1	96.7	30.6↓	94.3
w/o ToCo	33.9	96.6	28.7↓	92.4↓

Statistically significant drops at p-value < 0.05 are marked with ↓.

- Without typos correction training (namely, w/o ToCo), the performance of ToCoTR on both original and typoed queries is greatly affected.
- The adaption of hard negative mining can demonstrate why advanced retrievers with sophisticated training can outperform most typos-aware retrievers.

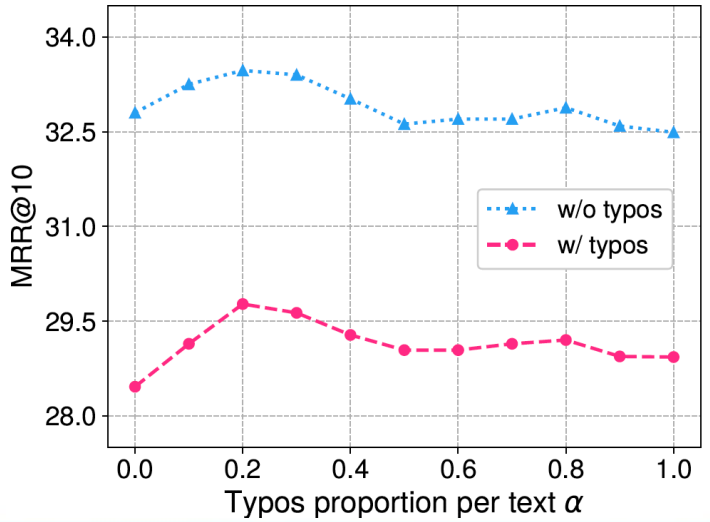
- Analysis on typos correction training

Joint or Two-stage?

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 ‡.

The impact of the number of typos (α).



The more typos injected, the worse performance becomes.

The proportion of typos per input text α is set as 0.2.

Results & Analysis

- 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 $\star$.

- Drop rate in different 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/-13.1%	23.8/-42.0%	84.2/-14.7%
RandDelete	**30.6**/-11.1%	**94.5**/-2.22%	22.7/-40.6%	86.2/-12.4%	24.7/-40.0%	86.0/-12.9%
RandSub	**30.1**/-12.3%	**93.6**/-3.17%	22.0/-42.6%	84.6/-14.0%	24.2/-41.1%	84.0/-14.9%
SwapNeighbor	**32.1**/-6.69%	**95.1**/-1.57%	21.6/-43.7%	83.5/-15.1%	22.9/-44.4%	82.5/-16.4%
SwapAdjacent	**30.9**/-10.1%	**94.0**/-2.68%	22.8/-40.6%	84.5/-14.1%	23.9/-42.0%	84.2/-14.7%

- ToCoTR can reach a smaller drop rate from original query set to various misspelled query variations.
- Different retrievers exhibit varying degrees of performance degradation with different simulated typos: ToCoTR with RandSub, coCondenser and SimLM with SwapNeighbor.

Conclusion

- 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 typed 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. It outperforms those with typos-aware training retrievers and even outperforms the combining solutions involving some advanced spell checkers for dense retrievals.

LREC-COLING 2024

Thanks

MLPD

四川大學 SICHUAN UNIVERSITY