Optimizing Test-Time Query Representations for Dense Retrieval

Mujeen Sung[1] **Jungsoo Park**[1] **Jaewoo Kang**[1] **Danqi Chen**[2] **Jinhyuk Lee**[3*]
Korea University[1] Princeton University[2] Google Research[3]
{mujeensung,jungsoo_park,kangj}@korea.ac.kr
danqic@cs.princeton.edu jinhyuklee@google.com

Abstract

Recent developments of dense retrieval rely on quality representations of queries and contexts from pre-trained query and context encoders. In this paper, we introduce TOUR (**T**est-Time **O**ptimization of **U**ery **R**epresentations), which further optimizes *instance-level* query representations guided by signals from test-time retrieval results. We leverage a cross-encoder re-ranker to provide fine-grained *pseudo labels* over retrieval results and iteratively optimize query representations with gradient descent. Our theoretical analysis reveals that TOUR can be viewed as a generalization of the classical Rocchio algorithm for pseudo relevance feedback, and we present two variants that leverage pseudo-labels as hard binary or soft continuous labels. We first apply TOUR on phrase retrieval with our proposed phrase re-ranker, and also evaluate its effectiveness on passage retrieval with an off-the-shelf re-ranker. TOUR greatly improves end-to-end open-domain question answering accuracy, as well as passage retrieval performance. TOUR also consistently improves direct re-ranking by up to 2.0% while running 1.3–2.4× faster with an efficient implementation.[1]

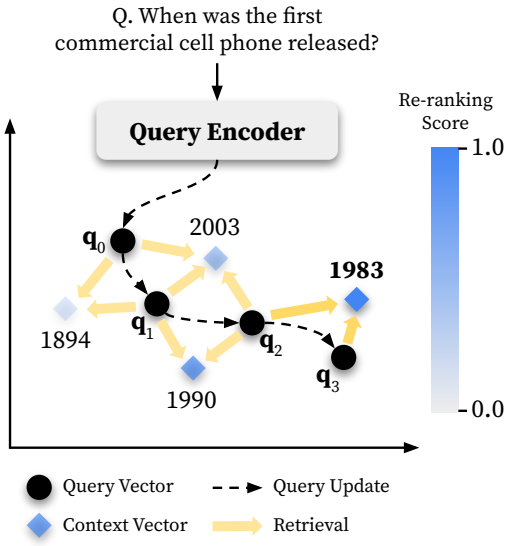

Figure 1: An overview of test-time optimization of query representations (TOUR). Given the initial representation of a test query $\mathbf{q}_0$, TOUR iteratively optimizes its representation (e.g., $\mathbf{q}_0 \rightarrow \mathbf{q}_1 \rightarrow \mathbf{q}_2 \rightarrow \mathbf{q}_3$) based on top-$k$ retrieval results. The figure shows how each query vector retrieves new context vectors and updates its representation to find the gold answer (e.g., 1983). Our cross-encoder re-ranker provides a relevance score for each top retrieval result making the query representation closer to the final answer.

1 Introduction

Recent progress in pre-trained language models gave birth to dense retrieval, which typically learns dense representations of queries and contexts in a contrastive learning framework. By overcoming the term mismatch problem, dense retrieval has been shown to be more effective than sparse retrieval in open-domain question answering (QA) (Lee et al., 2019; Karpukhin et al., 2020; Lee et al., 2021a) and information retrieval (Khattab and Zaharia, 2020; Xiong et al., 2020).

Dense retrieval often uses a dual encoder architecture, which enables the pre-computation of context representations while the query representations are directly computed from the trained encoder during inference. However, directly using trained query encoders often fails to retrieve the relevant context (Thakur et al., 2021; Sciavolino et al., 2021) as many test queries are unseen during training.

In this paper, we introduce TOUR, which further optimizes instance-level query representations at test time for dense retrieval. Specifically, we treat each test query as a single data point and iteratively optimize its representation. This resembles the query-side fine-tuning proposed for phrase retrieval (Lee et al., 2021a), which fine-tunes the query encoder over *training* queries in a new domain. Instead, we fine-tune query rep-

*Work partly done while visiting Princeton University.
[1]Our code is available at https://github.com/dmis-lab/TouR.

Findings of the Association for Computational Linguistics: ACL 2023, pages 5731–5746
July 9-14, 2023

resentations for each *test* query. Cross-encoders are known to exhibit better generalization ability in unseen distributions compared to dual encoders (Rosa et al., 2022). Accordingly, we leverage cross-encoder re-rankers (Nogueira and Cho, 2019; Fajcik et al., 2021) to provide *pseudo relevance labels* on intermediate retrieval results and then iteratively optimize query representations using gradient descent. For phrase retrieval, we also develop a cross-encoder phrase re-ranker, which has not been explored in previous studies.

We theoretically show that our framework can be viewed as a generalized version of the Rocchio algorithm for pseudo relevance feedback (PRF; Rocchio, 1971), which is commonly used in information retrieval to improve query representations. While most PRF techniques assume that the top-ranked results are equally pseudo-relevant, our method dynamically labels the top results and updates the query representations accordingly. We leverage our pseudo labels as either hard binary or soft continuous labels in two instantiations of our method, respectively. Lastly, to reduce computational overhead, we present an efficient implementation of TOUR, which significantly improves its runtime efficiency.

We apply TOUR on phrase (Lee et al., 2021a) and passage retrieval (Karpukhin et al., 2020) for open-domain QA. Experiments show that TOUR consistently improves performance in both tasks, even when the query distribution changes greatly. Specifically, TOUR improves the end-to-end open-domain QA accuracy by up to 10.7%, while also improving the accuracy of the top-20 passage retrieval by up to 8.3% compared to baseline retrievers. TOUR requires only a handful of top-k candidates to perform well, which enables TOUR to run up to 1.3–2.4× faster than the direct application of re-ranker with our efficient implementation while consistently improving the performance by up to 2.0%. The ablation study further shows the effectiveness of each component, highlighting the importance of fine-grained relevance signals.

2 Background

2.1 Dense Retrieval

Dense retrieval typically uses query and context encoders—$E_{\text{q}}(\cdot)$ and $E_{\text{c}}(\cdot)$—for representing queries and contexts, respectively (Lee et al., 2019; Karpukhin et al., 2020). In this work, we focus on improving phrase or passage retrievers for open-domain QA. The similarity of a query q and a context c is computed based on the inner product between their dense representations:

$$\text{sim}(q, c) = E_q(q)^\intercal E_c(c) = \mathbf{q}^\intercal \mathbf{c}. \quad (1)$$

Dense retrievers often use the contrastive learning framework to train encoders E_q and E_c. After training the encoders, top-k results are retrieved from a set of contexts $\mathcal{C}$:

$$\mathcal{C}^q_{1:k} = [c_1, \ldots, c_k] = \text{top-}k_{c \in \mathcal{C}} \text{sim}(q, c), \quad (2)$$

where the top-k operator returns a sorted list of contexts by their similarity score $\text{sim}(q, c)$ in descending order, i.e., $\text{sim}(q, c_1) \geq \cdots \geq \text{sim}(q, c_k)$. Dense retrievers aim to maximize the probability that a relevant context c^* exists (or is highly ranked) in the top results.

2.2 Query-side Fine-tuning

After training the query and context encoders, the context representations $\{\mathbf{c} \mid c \in \mathcal{C}\}$ are typically pre-computed for efficient retrieval while the query representations $\mathbf{q}$ are directly computed from the query encoder during inference. However, using the dense representations of queries as is often fails to retrieve relevant contexts, especially when the test query distribution is different from the one seen during training.

To mitigate the problem, Lee et al. (2021a) propose to fine-tune the query encoder on the retrieval results of training queries $\{q \mid q \in \mathcal{Q}_{\text{train}}\}$ over the entire corpus $\mathcal{C}$. For phrase retrieval (i.e., c denotes a phrase), they maximize the marginal likelihood of relevant phrases in the top-k results:

$$\mathcal{L}_{\text{query}} = -\sum_{q \in \mathcal{Q}_{\text{train}}} \log \sum_{c \in \mathcal{C}^q_{1:k}, c = c^*} P_k(c|q), \quad (3)$$

where $P_k(c|q) = \frac{\exp(\text{sim}(q,c))}{\sum_{i=1}^{k} \exp(\text{sim}(q,c_i))}$ and $c = c^*$ checks whether each context matches the gold context c^* or not. Note that c^* is always given for training queries. The query-side fine-tuning significantly improves performance and provides a means of efficient transfer learning when there is a query distribution shift. In this work, compared to training on entire training queries as in Eq. (3), we treat each test query $q \in \mathcal{Q}_{\text{test}}$ as a single data point to train on and optimize instance-level query representations at test time. This is in contrast to distillation-based passage retrievers (Izacard and Grave, 2020; Ren et al., 2021), which fine-tune the parameters of the retrievers directly on all training data by leveraging signals from cross-encoders.

2.3 Pseudo Relevance Feedback

Pseudo relevance feedback (PRF) techniques in information retrieval (Rocchio, 1971; Lavrenko and Croft, 2001) share a similar motivation to ours in that they refine query representations for a single test query. Unlike using the true relevance feedback provided by users (Baumgärtner et al., 2022), PRF relies on heuristic or model-based relevance feedback, which can be easily automated. Although most previous work uses PRF for sparse retrieval (Croft et al., 2010; Zamani et al., 2018; Li et al., 2018; Mao et al., 2021), recent work has begun to apply PRF for dense retrieval (Yu et al., 2021; Wang et al., 2021; Li et al., 2021).

PRF aims to improve the quality of the retrieval by updating the initial query representation from the query encoder (i.e., $E_q(q) = \mathbf{q}_0$):

$$\mathbf{q}_{t+1} \leftarrow g(\mathbf{q}_t, \mathcal{C}^{q_t}_{1:k}), \tag{4}$$

where g is an update function and $\mathbf{q}_t$ denotes the query representation after t-th updates over $\mathbf{q}_0$.

The classical Rocchio algorithm for PRF (Rocchio, 1971) updates the query representation as:

$$g(\mathbf{q}_t, \mathcal{C}^{q_t}_{1:k}) = \alpha\mathbf{q}_t + \beta\frac{1}{|\mathcal{C}_r|}\sum_{c_r \in \mathcal{C}_r}\mathbf{c}_r - \gamma\frac{1}{|\mathcal{C}_{nr}|}\sum_{c_{nr} \in \mathcal{C}_{nr}}\mathbf{c}_{nr}, \tag{5}$$

where $\mathcal{C}_r$ and $\mathcal{C}_{nr}$ denote *relevant* and *non-relevant* sets of contexts, respectively. α, β, and γ determine the relative contribution of the current query representation $\mathbf{q}_t$, relevant context representations $\mathbf{c}_r$, and non-relevant context representations $\mathbf{c}_{nr}$, respectively, when updating to $\mathbf{q}_{t+1}$. A common practice is to choose top-k' contexts as pseudo-relevant among top-k ($k' < k$), i.e., $\mathcal{C}_r = \mathcal{C}^{q_t}_{1:k'}$:

$$g(\mathbf{q}_t, \mathcal{C}^{q_t}_{1:k}) = \alpha\mathbf{q}_t + \beta\frac{1}{k'}\sum_{i=1}^{k'}\mathbf{c}_i - \gamma\frac{1}{k-k'}\sum_{i=k'+1}^{k}\mathbf{c}_i. \tag{6}$$

In this work, we theoretically show that our test-time query optimization is a generalization of the Rocchio algorithm. While Eq. (6) treats the positive (or negative) contexts equally, we use cross-encoder re-rankers (Nogueira and Cho, 2019) to provide fine-grained pseudo labels and optimize the query representations with gradient descent.

3 Methodology

In this section, we provide an overview of our method (§3.1) and its two instantiations (§3.2, §3.3). We also introduce a relevance labeler for phrase retreival (§3.4) and simple techniques to improve efficiency of TOUR (§3.5).

3.1 Optimizing Test-time Query Representations

We propose TOUR (Test-Time Optimization of Query Representations), which optimizes query representations at the instance level. In our setting, the query and context encoders are fixed after training, and we optimize the query representations solely based on their retrieval results. Figure 1 illustrates an overview of TOUR.

First, given a single test query $q \in \mathcal{Q}_{\text{test}}$, we use a cross-encoder re-ranker $\phi(\cdot)$ to provide a score of how relevant each of the top-k contexts $c \in \mathcal{C}^q_{1:k}$ is with respect to a query:

$$s = \phi(q, c), \tag{7}$$

where $\phi(\cdot)$ is often parameterized with a pre-trained language model, which we detail in §3.4. Compared to simply setting top-k' results as pseudo-positive in PRF, using cross-encoders enables more fine-grained judgments of relevance over the top results. In addition, it allows us to label results for *test* queries as well without access to the gold label c^*.

3.2 TOUR with Hard Labels : $\text{TOUR}_{\text{hard}}$

First, we explore using the scores from the cross-encoder labeler ϕ and selecting a set of pseudo-positive contexts $\mathcal{C}^q_{\text{hard}} \subset \mathcal{C}^q_{1:k}$ defined as the smallest set such that:

$$P_k(\tilde{c} = c^* | q, \phi) = \frac{\exp(\phi(q, \tilde{c})/\tau)}{\sum_{i=1}^{k}\exp(\phi(q, c_i)/\tau)}$$
$$\sum_{\tilde{c} \in \mathcal{C}^q_{\text{hard}}} P_k(\tilde{c} = c^* | q, \phi) \geq p, \tag{8}$$

where τ is a temperature parameter and $\tilde{c} \in \mathcal{C}^q_{\text{hard}}$ denotes a pseudo-positive context selected by ϕ. Intuitively, we choose the smallest set of contexts as $\mathcal{C}^q_{\text{hard}}$ whose marginal relevance with respect to a query under ϕ is larger than the threshold p. This is similar to Nucleus Sampling for stochastic decoding (Holtzman et al., 2020).

Then, TOUR optimizes the query representation with the gradient descent algorithm based on the

relevance judgment $\mathcal{C}^q_{\text{hard}}$ made by ϕ:

$$\mathcal{L}_{\text{hard}}(q, \mathcal{C}^q_{1:k}) = -\log \sum_{\tilde{c} \in \mathcal{C}^q_{\text{hard}}} P_k(\tilde{c}|q), \quad (9)$$

where $P_k(\tilde{c}|q) = \frac{\exp(\text{sim}(q,\tilde{c}))}{\sum_{i=1}^{k} \exp(\text{sim}(q,c_i))}$. Similar to the query-side fine-tuning in Eq. (3), we maximize the marginal likelihood of (pseudo) positive contexts $\mathcal{C}^q_{\text{hard}}$. We denote this version as TOUR$_{\text{hard}}$. Unlike query-side fine-tuning that updates the model parameters of $E_q(\cdot)$, we directly optimize the query representation $\mathbf{q}$ itself. TOUR$_{\text{hard}}$ is also an instance-level optimization over a single test query $q \in \mathcal{Q}_{\text{test}}$ without access to the gold label c^*.

For optimization, we use gradient descent:

$$\mathbf{q}_{t+1} \leftarrow \mathbf{q}_t - \eta \frac{\partial \mathcal{L}_{\text{hard}}(\mathbf{q}_t, \mathcal{C}^{q_t}_{1:k})}{\partial \mathbf{q}_t}, \quad (10)$$

where η denotes the learning rate for gradient descent and the initial query representation is used as $\mathbf{q}_0$. Applying gradient descent over the test queries shares the motivation with dynamic evaluation for language modeling (Krause et al., 2019), but we treat each test query independently unlike the series of tokens for the evaluation corpus of language modeling. For each iteration, we perform a single step of gradient descent followed by another retrieval with $\mathbf{q}_{t+1}$ to update $\mathcal{C}^{q_t}_{1:k}$ into $\mathcal{C}^{q_{t+1}}_{1:k}$.

Relation to the Rocchio algorithm Eq. (10) could be viewed as performing PRF by setting the update function $g(\mathbf{q}_t, \mathcal{C}^{q_t}_{1:k}) = \mathbf{q}_t - \eta \frac{\partial \mathcal{L}_{\text{hard}}(\mathbf{q}_t, \mathcal{C}^{q_t}_{1:k})}{\partial \mathbf{q}_t}$. In fact, our update rule Eq. (10) is a generalized version of the Rocchio algorithm as shown below:

$$\begin{aligned} &g(\mathbf{q}_t, \mathcal{C}^{q_t}_{1:k}) \\ &= \mathbf{q}_t + \eta \sum_{\tilde{c}} P(\tilde{c}|q_t)(1 - P_k(\tilde{c}|q_t))\tilde{\mathbf{c}} \\ &\quad - \eta \sum_{\tilde{c}} \Big[P(\tilde{c}|q_t) \sum_{c \in \mathcal{C}^{q_t}_{1:k}, c \neq \tilde{c}} P_k(c|q_t)\mathbf{c} \Big], \end{aligned} \quad (11)$$

where $\tilde{c} \in \mathcal{C}^{q_t}_{\text{hard}}$ and $P(\tilde{c}|q_t) = \frac{\exp(\text{sim}(q_t,\tilde{c}))}{\sum_{\tilde{c}'} \exp(\text{sim}(q_t,\tilde{c}'))}$ (proof in Appendix A). Although our update rule seems to fix α in Rocchio to 1, it can be dynamically changed by applying weight decay during gradient descent, which sets $\alpha = 1 - \eta\lambda_{\text{decay}}$ multiplied by $\mathbf{q}_t$. Then, the equality between Eq. (6) and Eq. (11) holds when $\mathcal{C}^{q_t}_{\text{hard}} = \mathcal{C}^{q_t}_{1:k'}$ with $P_k(c|q_t)$ being equal for all $c \in \mathcal{C}^{q_t}_{1:k}$, namely $P_k(c|q_t) = 1/k$. This reflects that the Rocchio algorithm treats all top-k' results equally (i.e., $P(\tilde{c}|q_t) = 1/k'$). Then, $\beta = \gamma = \eta\frac{k-k'}{k}$ holds (Appendix C).

In practice, $\mathcal{C}^{q_t}_{\text{hard}}$ would be different from $\mathcal{C}^{q_t}_{1:k'}$ if some re-ranking happens by ϕ. Also, each pseudo-positive context vector $\tilde{\mathbf{c}}$ in the second term of the RHS of Eq. (11) has a different weight. The contribution of $\tilde{\mathbf{c}}$ is maximized when it has a larger probability mass $P(\tilde{c}|q_t)$ among the pseudo-positive contexts, but a smaller probability mass $P_k(\tilde{c}|q_t)$ among the top-k contexts; this is desirable since we want to update $\mathbf{q}_t$ a lot when the initial ranking of pseudo-positive context in top-k is low. For instance, if there is a single pseudo-positive context $\tilde{c}$ (i.e., $P(\tilde{c}|q_t)) = 1$) ranked at the bottom of top-k with a large margin with top-1 (i.e., $P_k(\tilde{c}|q_t) = 0$), then $P(\tilde{c}|q_t)(1 - P_k(\tilde{c}|q_t)) = 1$ is maximized.

3.3 TOUR with Soft Labels : TOUR$_{\text{soft}}$

From Eq. (11), we observe that it uses pseudo-positive contexts $\mathcal{C}^{q_t}_{\text{hard}}$ sampled by the cross-encoder labeler ϕ, but the contribution of $\tilde{\mathbf{c}}$ (the second term in RHS) does not directly depend on the scores from ϕ. The scores are only used to make a hard decision in pseudo-positive contexts. Another version of TOUR uses the normalized scores of a cross-encoder over the retrieved results as soft labels. We can simply change the maximum marginal likelihood objective in Eq. (9) to reflect the scores from ϕ in g. Specifically, we change Eq. (9) to minimize Kullback-Leibler (KL) divergence loss as follows:

$$\begin{aligned} &\mathcal{L}_{\text{soft}}(\mathbf{q}_t, \mathcal{C}^{q_t}_{1:k}) = \\ &\quad - \sum_{i=1}^{k} P(c_i|q_t, \phi) \log \frac{P_k(c_i|q_t)}{P(c_i|q_t, \phi)}, \end{aligned} \quad (12)$$

where $P(c_i|q_t, \phi) = P(c_i = c^*|q_t, \phi)$ defined in Eq. (8). We call this version TOUR$_{\text{soft}}$. The update rule g for TOUR$_{\text{soft}}$ changes as follows:

$$\begin{aligned} &g(\mathbf{q}_t, \mathcal{C}^{q_t}_{1:k}) \\ &= \mathbf{q}_t + \eta \sum_{i=1}^{k} P(c_i|q_t, \phi)\mathbf{c}_i - \eta \sum_{i=1}^{k} P_k(c_i|q_t)\mathbf{c}_i. \end{aligned} \quad (13)$$

Eq. (13) shows that $\mathbf{q}_{t+1}$ reflects $\mathbf{c}_i$ weight-averaged by the cross-encoder (i.e., $P(c_i|q_t, \phi)$) while removing $\mathbf{c}_i$ weight-averaged by the current retrieval result (i.e., $P_k(c_i|q_t)$) (proof in Appendix B).

3.4 Relevance Labeler for Phrase Retrieval

In the previous section, we used a cross-encoder reranker ϕ to provide a relevance score s_i over

a pair of a query q and a context c. While it is possible to use an off-the-shelf re-ranker (Fajcik et al., 2021) for passage retrieval, no prior work has introduced a re-ranker for phrase retrieval (Lee et al., 2021b). In this section, we introduce a simple and accurate phrase re-ranker for TOUR.

Inputs for re-rankers For phrase retrieval, sentences containing each retrieved phrase are considered as contexts, following Lee et al. (2021b). For each context, we also prepend the title of its document and use it as our context for re-rankers. To train our re-rankers, we first construct a training set from the retrieved contexts of the phrase retriever given a set of training queries $\mathcal{Q}_{\text{train}}$. Specifically, from the top retrieved contexts $\mathcal{C}_{1:k}$ for every $q \in \mathcal{Q}_{\text{train}}$, we sample one positive context c_q^+ and one negative context c_q^-. In open domain QA, it is assumed that a context that contains a correct answer to each q is relevant (positive). Our re-ranker is trained on a dataset $\mathcal{D}_{\text{train}} = \{(q, c_q^+, c_q^-) | q \in \mathcal{Q}_{\text{train}}\}$.

Architecture We use the RoBERTa-large model (Liu et al., 2019) as the base model for our re-ranker. Given a pre-trained LM $\mathcal{M}$, the cross-encoder re-ranker ϕ outputs a score of a context being relevant:

$$s = \phi(q, c) = \mathbf{w}^\top \mathcal{M}(q \oplus c)[\text{CLS}] \quad (14)$$

where $\{\mathcal{M}, \mathbf{w}\}$ are the trainable parameters and $\oplus$ denotes a concatenation of q and c using a [SEP] token. Since phrase retrievers return both phrases and their contexts, we use special tokens [S] and [E] to mark the retrieved phrases within the contexts.

Re-rankers are trained to maximize the probability of a positive context c_q^+ for every $(q, c_q^+, c_q^-) \in \mathcal{D}_{\text{train}}$. We use the binary cross-entropy loss defined over the probability $P^+ = \frac{\exp(\mathbf{h}^+)}{\exp(\mathbf{h}^+)+\exp(\mathbf{h}^-)}$ where $\mathbf{h}^+ = \phi(q, c_q^+)$ and $\mathbf{h}^- = \phi(q, c_q^-)$. We pre-train ϕ on reading comprehension datasets (Rajpurkar et al., 2016; Joshi et al., 2017; Kwiatkowski et al., 2019), which helped improve the quality of ϕ. For the ablation study of our phrase re-rankers, see Appendix D for details.

Score aggregation After running TOUR, aggregating the reranking scores with the retreival scores provides consistent improvement. Specifically, we linearly interpolate the similarity score $\text{sim}(q, c_i)$ with the re-ranking score s_i and use this to obtain the final results: $\lambda s_i + (1 - \lambda)\text{sim}(q, c_i)$.

3.5 Efficient Implementation of TOUR

TOUR aims to improve the recall of gold candidates by iteratively searching with updated query representations. However, it has high computational complexity, since it needs to label top-k retrieval results with a cross-encoder and perform additional retrieval. To minimize the additional time complexity, we perform up to $t = 3$ iterations with early stopping conditions. Specifically, at every iteration of $\text{TOUR}_{\text{hard}}$, we stop when the top-1 retrieval result is pseudo-positive, i.e., $c_1 \in \mathcal{C}_{\text{hard}}^{q_t}$. When using $\text{TOUR}_{\text{soft}}$, we stop iterating when the top-1 retrieval result has the highest relevance score. Additionally, we cache $\phi(q, c_i)$ for each query to skip redundant computation.

4 Experiments

We test TOUR on multiple open-domain QA datasets. Specifically, we evaluate its performance on phrase retrieval and passage retrieval.

4.1 Datasets

We mainly use six open-domain QA datasets: Natural Questions (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), WebQuestions (Berant et al., 2013), CuratedTrec (Baudiš and Šedivỳ, 2015), SQuAD (Rajpurkar et al., 2016), and EntityQuestions (Sciavolino et al., 2021). Following previous works, Entity Questions is only used for testing. See statistics in Appendix E.

4.2 Open-domain Question Answering

For end-to-end open-domain QA, we use phrase retrieval (Seo et al., 2019; Lee et al., 2021a) for TOUR, which directly retrieves phrases from the entire Wikipedia using a phrase index. Since a single-stage retrieval is the only component in phrase retrieval, it is easy to show how its open-domain QA performance can be directly improved with TOUR. We use DensePhrases (Lee et al., 2021a) for our base phrase retrieval model and train a cross-encoder labeler as described in §3.4. We report exact match (EM) for end-to-end open-domain QA. We use $k = \{10, 40\}$ for our phrase re-ranker and $k = \{10, 20\}$ for TOUR on open-domain QA while $k = 10$ is used for both whenever it is omitted. For the implementation details of TOUR, see Appendix D.

Baselines Many open-domain QA models take the retriever-reader approach (Chen et al., 2017; Lee et al., 2019; Izacard and Grave, 2021; Singh

Model	Top-k	s/q (↓)	NQ	TriviaQA	WQ	TREC	SQuAD
Retriever + Extractive Reader							
$\text{DPR}_{\text{multi}}$ (Karpukhin et al., 2020)			41.5	56.8	42.4	49.4	24.1
+ Re-ranker (Iyer et al., 2021)	5	1.21	43.1 ↑1.6	59.3 ↑2.5	44.4 ↑2.0	49.3 ↓0.1	-
GAR (Mao et al., 2021)			41.8	62.7	-	-	-
$\text{DPR}_{\text{multi}}$ (large)			44.6	60.9	44.8	53.5	-
+ Re-ranker	5	>1.21*	45.5 ↑0.9	61.7 ↑0.8	**45.9** ↑**1.1**	**55.3** ↑**1.8**	-
$\text{ColBERT-QA}_{\text{large}}$ (Khattab et al., 2021)			47.8	**70.1**	-	-	**54.7**
$\text{UnitedQA-E}_{\text{large}}$			**51.8**	68.9	-	-	-
Retriever-only							
$\text{DensePhrases}_{\text{multi}}$ (Lee et al., 2021a)			41.6	56.3	41.5	53.9	34.5
+ $\text{PRF}_{\text{Rocchio}}$	10	0.09	41.6 0.0	56.5 ↑0.2	41.7 ↑0.2	54.0 ↑0.1	34.9 ↑0.4
+ Phrase re-ranker (Ours)	10	0.24	47.0 ↑5.4	65.4 ↑9.1	45.9 ↑4.4	60.5 ↑6.6	43.1 ↑8.6
+ Phrase re-ranker (Ours)	40	1.04	46.5 ↑4.9	66.0 ↑9.7	46.3 ↑4.8	61.5 ↑7.6	45.3 ↑10.8
+ $\text{TouR}_{\text{hard}}$ (Ours)	10	0.44	**48.6** ↑**7.0**	66.4 ↑10.1	46.1 ↑4.6	62.0 ↑8.1	45.2 ↑10.7
+ $\text{TouR}_{\text{hard}}$ (Ours)	20	0.78	47.9 ↑6.3	**66.8** ↑**10.5**	**46.9** ↑**5.4**	62.5 ↑8.6	**46.4** ↑**11.9**
+ $\text{TouR}_{\text{soft}}$ (Ours)	10	0.43	47.9 ↑6.3	66.5 ↑10.2	46.3 ↑4.8	**63.1** ↑**9.2**	44.9 ↑10.4
+ $\text{TouR}_{\text{soft}}$ (Ours)	20	0.78	47.6 ↑6.0	66.6 ↑10.3	**46.9** ↑**5.4**	62.5 ↑8.6	46.0 ↑11.5

Table 1: Open-domain QA results. We report exact match (EM) on each test set. s/q denotes the average latency of a single query in seconds, which includes the latency of $\text{DPR}_{\text{multi}}$ or $\text{DensePhrases}_{\text{multi}}$. For re-ranking and PRF-based methods, we denote the improvement from their base retrievers in ↑x.x. Best performance is denoted in **bold**. *: We could not measure the exact latency of $\text{DPR}_{\text{multi}}$ (large) as its checkpoint has not been released.

et al., 2021). As our baselines, we report extractive open-domain QA models, which is a fair comparison with retriever-only (+ re-ranker) models whose answers are always extractive. For re-ranking baselines of retriever-reader models, we report ReConsider (Iyer et al., 2021), which re-ranks the outputs of DPR + BERT. For a PRF baseline, GAR (Mao et al., 2021), which uses context generation models for augmenting queries in BM25, is reported.

Results Table 1 shows the results on the five open-domain QA datasets in the in-domain evaluation setting where all models use the training sets of each dataset they are evaluated on. First, we observe that using our phrase re-ranker largely improves the performance of $\text{DensePhrases}_{\text{multi}}$. Compared to adding a re-ranker on the retriever-reader model ($\text{DPR}_{\text{multi}}$ + Re-ranker by Iyer et al., 2021), our phrase re-ranking approach performs 5× faster with a larger top-k due to the efficient retriever-only method. Furthermore, the performance gain is significantly larger possibly due to the high top-k accuracy of phrase retrievers. Unlike using the Rocchio algorithm, using $\text{TouR}_{\text{hard}}$ or $\text{TouR}_{\text{soft}}$ greatly improves the performance of the base retriever. Compared to our phrase re-ranker$_{k=40}$, $\text{TouR}_{\text{hard},k=20}$ runs 1.3× faster as well as outperforming it by up to 2.0%. Even $\text{TouR}_{\text{hard},k=10}$ often outperforms re-ranker$_{k=40}$ with 2.4× faster inference. For this task, $\text{TouR}_{\text{hard}}$ and $\text{TouR}_{\text{soft}}$ work similarly with exceptions on NQ and TREC.

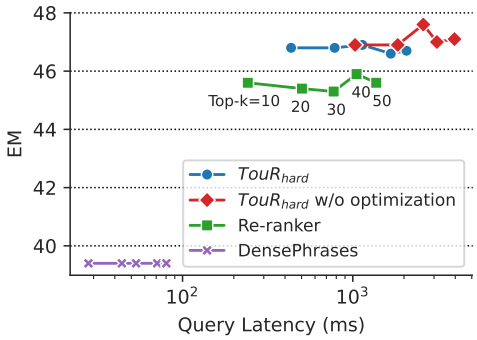

Figure 2: Query latency (ms) vs. open-domain QA performance (EM) of different models on the NQ development set. Query latency is controlled by varying the top-k value incrementally. $\text{TouR}_{\text{hard}}$ w/o optimization: $\text{TouR}_{\text{hard}}$ without the efficient implementation in §3.5.

Latency vs. performance Figure 2 compares the query latency and performance of TouR and other baselines on the NQ development set. We vary the top-k value from 10 to 50 by 10 (left to right) to visualize the trade-off between latency and performance. The result shows that TouR with only top-10 is better and faster than the re-ranker with the best top-k. Specifically, $\text{TouR}_{\text{hard},k=10}$ outperforms re-ranker$_{k=40}$ by 1.0% while being 2.5× faster. This shows that TouR requires a less number of retrieval results to perform well, compared to a re-ranker model that often requires a larger k.

Model	Top-k	s/q (↓)	*Training* NQ	*Unseen Query Distribution* TRIVIAQA	WQ	TREC	SQUAD	ENTITYQ[2]
DPR_{NQ}* (Karpukhin et al., 2020)			39.4	29.4	-	-	0.1	-
$\text{DensePhrases}_{\text{NQ}}$ (Lee et al., 2021a)			40.8	33.4	23.8	33.6	15.4	22.4
+ Phrase re-ranker (Ours)	10	0.24	45.4 ↑4.6	40.9 ↑7.5	26.6 ↑2.8	37.8 ↑4.2	20.2 ↑4.8	26.8 ↑4.4
+ Phrase re-ranker (Ours)	40	1.04	44.5 ↑3.7	41.7 ↑8.3	26.4 ↑2.6	37.8 ↑4.2	21.4 ↑6.0	27.1 ↑4.7
+ $\text{TOUR}_{\text{hard}}$ (Ours)	10	0.44	**47.0 ↑6.2**	42.6 ↑9.2	27.7 ↑3.9	38.3 ↑4.7	21.5 ↑6.1	27.9 ↑5.5
+ $\text{TOUR}_{\text{hard}}$ (Ours)	20	0.78	46.5 ↑5.7	**42.9 ↑9.5**	**28.2 ↑4.4**	**39.8 ↑6.2**	**22.1 ↑6.7**	**28.3 ↑5.9**
+ $\text{TOUR}_{\text{soft}}$ (Ours)	10	0.43	46.2 ↑5.4	42.5 ↑9.1	27.4 ↑3.6	38.2 ↑4.6	21.2 ↑5.8	27.6 ↑5.2
+ $\text{TOUR}_{\text{soft}}$ (Ours)	20	0.78	45.7 ↑4.9	42.7 ↑9.3	27.7 ↑3.9	**39.8 ↑6.2**	21.7 ↑6.3	27.9 ↑5.5

Table 2: Open-domain QA results under query distribution shift. All retrievers and re-rankers are trained on NaturalQuestions and evaluated on unseen query distributions. EM is reported on each test set. *: results obtained from the official implementation, which does not support running end-to-end QA on WebQuestions, CuratedTREC, and EntityQuestions. ↑x.x shows EM improvement from $\text{DensePhrases}_{\text{NQ}}$.

Model	NQ (Acc@20/100)	TRIVIAQA (Acc@20/100)	ENTITYQ† (Acc@20/100)
$\text{DensePhrases}_{\text{multi}}$	79.8 / 86.0	81.6 / 85.8	61.0 / 71.2
+ Re-ranker (Fajcik et al., 2021)	83.2 / 86.0 ↑3.4	83.0 / 85.8 ↑1.4	65.3 / 71.2 ↑4.3
+ $\text{TOUR}_{\text{hard}}$ (Ours)	84.0 / 86.9 ↑4.2	**83.2 / 86.1 ↑1.6**	**66.2 / 72.4 ↑5.2**
+ $\text{TOUR}_{\text{soft}}$ (Ours)	**84.2 / 87.0 ↑4.4**	**83.2 / 86.1 ↑1.6**	**66.2 / 72.4 ↑5.2**
$\text{DPR}_{\text{multi}}$	79.4 / 86.5	79.0 / 84.8	57.9 / 70.8
+ Re-ranker (Fajcik et al., 2021)	83.6 / 86.5 ↑4.2	**81.6** / 84.8 ↑2.6	64.4 / 70.8 ↑6.5
+ $\text{TOUR}_{\text{hard}}$ (Ours)	84.0 / 87.0 ↑4.6	81.5 / 84.9 ↑2.5	65.6 / 71.9 ↑7.7
+ $\text{TOUR}_{\text{soft}}$ (Ours)	**84.2 / 87.2 ↑4.8**	**81.6 / 85.1 ↑2.6**	**66.2 / 72.5 ↑8.3**

Table 3: Passage retrieval results. We report Acc@20 / Acc@100 (%) on each test set. Each retriever (and re-ranker) is trained on multiple open-domain QA datasets described in §4.1, which makes Natural Questions and TriviaQA in-domain evaluation and leaves EntityQuestions as out-of-domain evaluation. We denote improvement in Acc@20 from $\text{DensePhrases}_{\text{multi}}$ or $\text{DPR}_{\text{multi}}$ in ↑x.x. †: unseen query distribution.

Query distribution shift In Table 2, we show open-domain QA results under query distribution shift from the training distribution. Compared to $\text{DensePhrases}_{\text{multi}}$ in Table 1, which was trained on all five open-domain QA datasets, we observe huge performance drops on unseen query distributions when using DPR_{NQ} and $\text{DensePhrases}_{\text{NQ}}$. DPR_{NQ} seems to suffer more (e.g., 0.1 on SQuAD) since both of its retriever and reader were trained on NQ, which exacerbates the problem when combined.

On the other hand, using TOUR largely improves the performance of $\text{DensePhrases}_{\text{NQ}}$ on many unseen query distributions even though all of its component were still trained on NQ. Specifically, $\text{TOUR}_{\text{hard},k=20}$ gives 6.5% improvement on average across different query distributions, which easily outperforms our phrase re-ranker$_{k=40}$. Interestingly, $\text{TOUR}_{\text{hard}}$ consistently performs better than $\text{TOUR}_{\text{soft}}$ in this setting, which requires more investigation in the future.

4.3 Passage Retrieval

We test TOUR on the passage retrieval task for open-domain QA. We use DPR as a passage retriever and DensePhrases as a phrase-based passage retriever (Lee et al., 2021b). In this experiment, we use an off-the-shelf passage re-ranker (Fajcik et al., 2021) to show how existing re-rankers can serve as a pseudo labeler for TOUR. We report the top-k retrieval accuracy, which is 1 when the answers exist in top-k retrieval results. For passage retrieval, we use $k = 100$ for both the re-ranker and TOUR due to the limited resource budget.

Results Table 3 shows the results of passage retrieval for open-domain QA. We find that using TOUR consistently improves the passage retrieval accuracy. Under the query distribution shift similar to Table 2, $\text{DPR}_{\text{multi}}$ + $\text{TOUR}_{\text{soft}}$ improves the original DPR by 8.3% and advances the off-the-shelf re-ranker by 1.8% on EntityQuestions (Acc@20). Notably, Acc@100 always improves with TOUR,

		Overlap		
NQ	Total	Query	$\text{Answer}_{\text{only}}$	None
$\text{DensePhrases}_{\text{multi}}$	41.3	63.3	33.7	23.9
Re-ranker (Ours)	46.8	66.7	39.0	31.0
$\text{TOUR}_{\text{hard}}$ (Ours)	**48.6**	**70.1**	**40.3**	**33.7**
TRIVIAQA				
$\text{DensePhrases}_{\text{multi}}$	53.8	76.5	46.2	32.6
Re-ranker (Ours)	62.8	82.1	60.3	41.5
$\text{TOUR}_{\text{hard}}$ (Ours)	**63.8**	**83.6**	**62.3**	**42.2**
WQ				
$\text{DensePhrases}_{\text{multi}}$	41.5	70.8	39.5	27.5
Re-ranker (Ours)	45.9	**73.4**	**48.5**	31.5
$\text{TOUR}_{\text{hard}}$ (Ours)	**46.2**	70.1	**48.5**	**33.0**

Table 4: Open-domain QA results on train-test overlap splits by Lewis et al. (2021). **Query** overlap denotes test queries that are paraphrases of training queries. $\textbf{Answer}_{\textbf{only}}$ overlap denotes test queries that have answers present in training data, while their queries are not overlapping with any training queries. **None** overlap denotes test queries without any query or answer overlap with training data. We report EM on each split.

	NQ
$\text{DensePhrases}_{\text{NQ}}$	42.4
$\text{DensePhrases}_{\text{NQ}}$ + $\text{TOUR}_{\text{hard}}$	**48.4**
$\mathcal{C}^q_{\text{hard}} \Rightarrow \mathcal{C}_{1:k'}$ $(k'=3)$	46.1
SGD $\Rightarrow$ interpolation $(\beta = 0.3)$	48.2
$\lambda = 0.1 \Rightarrow \lambda = 0$	48.1
$\lambda = 0.1 \Rightarrow \lambda = 1$	48.0
$\text{TOUR}_{\text{hard}} \Rightarrow \text{TOUR}_{\text{soft}}$	47.7

Table 5: Ablation study of $\text{TOUR}_{\text{hard}}$ on the Natural Questions (NQ) development set. We report EM for end-to-end open-domain QA.

which is not possible for re-rankers since they do not update the top retrieval results. Unlike the phrase retrieval task, we observe that $\text{TOUR}_{\text{soft}}$ is a slightly better option than $\text{TOUR}_{\text{hard}}$ on this task.

5 Analysis

5.1 Train-Test Overlap Analysis

Open-domain QA datasets often contain semantically overlapping queries and answers between training and test sets (Lewis et al., 2021), which overestimates the generalizability of QA models. Hence, we test our models on train-test overlap splits provided by Lewis et al. (2021). Table 4 shows that TOUR consistently improves the performance of test queries that do not overlap with training data (i.e., None). Notably, on WebQuestions, while the performance on the none overlap split has been improved by 1.5% from the re-ranker, the performance on query overlap is worse than the re-ranker since unnecessary exploration is often performed on overlapping queries. Our finding on the effectiveness of query optimization is similar to that of Mao et al. (2021), while our approach often improves performance on query overlap cases.

[2]While the passage retrieval accuracy is mostly reported for Entity Questions (Ram et al., 2022; Lewis et al., 2022), we also report EM for open-domain QA.

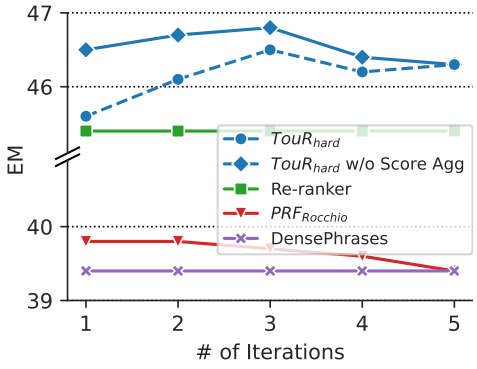

Figure 3: Effect of multiple iterations in $\text{TOUR}_{\text{hard}}$ and $\text{PRF}_{\text{Rocchio}}$. We report open-domain QA EM on the Natural Questions development set. Score Agg: score aggregation between the re-ranker and the retriever. Note that the performance of original DensePhrases and Re-ranker is not affected by iterations.

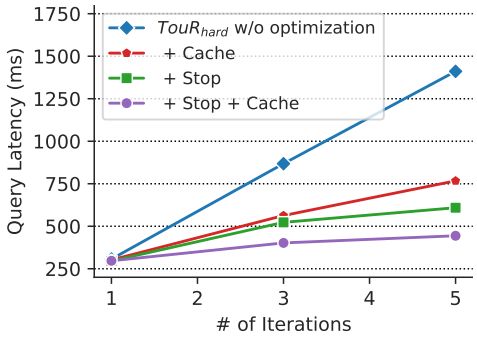

Figure 4: Ablation study of efficient implementation of $\text{TOUR}_{\text{hard}}$. Latency is reported for different numbers of iterations. Cache: caching $\phi(q, c_i)$ for every iteration. Stop: applying the stop condition of $c_1 \in \mathcal{C}^{q_t}_{\text{hard}}$.

5.2 Ablation Study

Table 5 shows an ablation study of $\text{TOUR}_{\text{hard}}$ on end-to-end open-domain QA. We observe that using fine-grained relevance signals generated by our phrase re-ranker (i.e., $\mathcal{C}^q_{\text{hard}}$) is significantly more effective than simply choosing top-k' as relevance

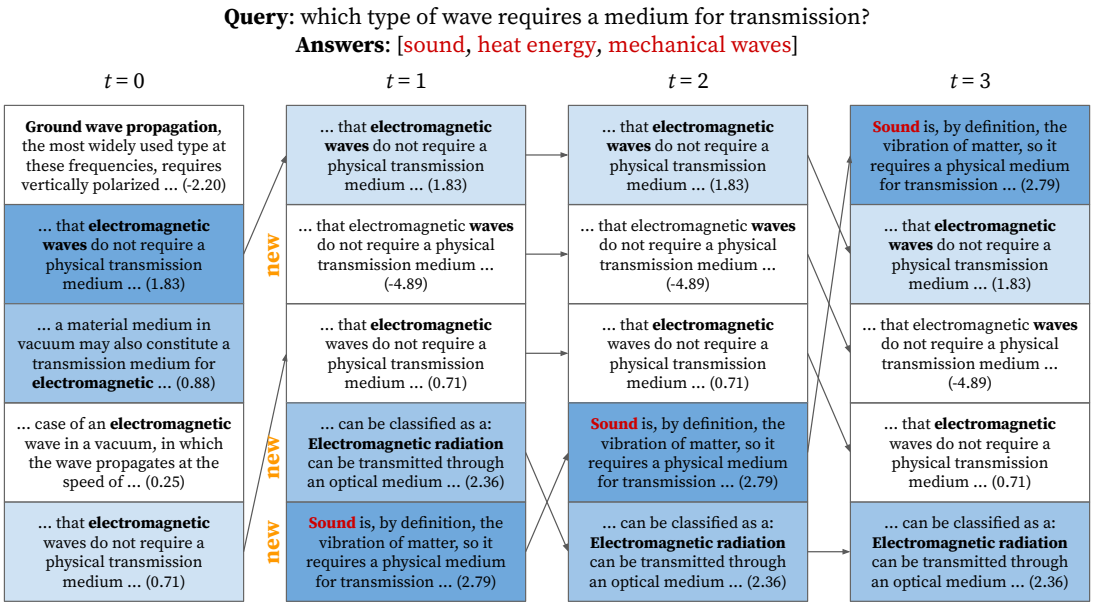

Figure 5: A sample prediction of TOUR$_{\text{hard}}$ from Natural Questions. For every t-th iteration of TOUR$_{\text{hard}}$, we show the top 5 phrases (denoted in bold) retrieved from DensePhrases along with their passages. The score s_i from the cross-encoder labeler ϕ is shown in each parenthesis. $t = 0$ denotes initial retrieval results. When $t = 1$, TOUR$_{\text{hard}}$ obtains three new results and the correct answer "Sound" becomes the top-1 prediction at $t = 3$.

signals (i.e., $\mathcal{C}_{1:k'}$). Using SGD or aggregating the final scores between the retriever and the re-ranker gives additional improvement.

Figure 3 shows the effect of multiple iterations in TOUR$_{\text{hard}}$ compared to the Rocchio algorithm. While PRF$_{\text{Rocchio}}$ with $t = 1$ achieves slightly better performance than DensePhrases, it shows a diminishing gain with a larger number of iterations. In contrast, the performance of TOUR$_{\text{hard}}$ benefits from multiple iterations until $t = 3$. Removing the score aggregation between the retriever and the re-ranker (i.e., $\lambda = 0$) causes a performance drop, but it quickly recovers with a larger t.

Efficient implementation Simple techniques introduced in §3.5 such as early stopping and caching significantly reduce the run-time of TOUR. Figure 4 summarizes the effect of optimization techniques to improve efficiency of TOUR. Without each technique, the latency increases linearly with the number of iterations. By adding the caching mechanism for ϕ and the stop condition of $c_1 \in \mathcal{C}^{q_t}_{\text{hard}}$, the latency is greatly reduced.

Prediction sample Figure 5 shows a sample prediction of TOUR. We use DensePhrases$_{\text{multi}}$ + TOUR$_{\text{hard}}$ with $k = 10$, from which the top-5 results are shown. While the initial result at $t = 0$ failed to retrieve correct answers in the top-10, the next round of TOUR$_{\text{hard}}$ gives new results including the correct answer, which were not retrieved before. As the iteration continues, the correct answer starts to appear in the top retrieval results, and becomes the top-1 at $t = 3$.

6 Conclusion

In this paper, we propose TOUR, which iteratively optimizes test query representations for dense retrieval. Specifically, we optimize instance-level query representations at test time using the gradient-based optimization method over the top retrieval results. We use cross-encoder re-rankers to provide pseudo labels where our simple re-ranker or off-the-shelf re-rankers can be used. We theoretically show that gradient-based optimization provides a generalized version of the Rocchio algorithm for pseudo relevance feedback, which leads us to develop different variants of TOUR. Experiments show that our test-time query optimization largely improves the retrieval accuracy on multiple open-domain QA datasets in various settings while being more efficient than traditional re-ranking methods.

Limitations

In this paper, we focus on the end-to-end accuracy and passage retrieval accuracy for open-domain QA. We have also experimented on the BEIR benchmark (Thakur et al., 2021) to evaluate our method in the zero-shot document retrieval task. Overall, we obtained 48.1% macro-averaged NDCG@10 compared to 47.8% by the re-ranking method. For some tasks, TOUR obtains significant improvements with a pre-trained document retriever (Hofstätter et al., 2021). For example, TOUR improves the baseline retriever by 11.6% and 23.8% NDCG@10 on BioASQ and TREC-COVID, respectively, while also outperforming the re-ranker by 2.1% and 2.4% NDCG@10. We plan to better understand why TOUR performs better specifically on these tasks and further improve it.

TOUR also requires a set of validation examples for hyperparameter selection. While we only used in-domain validation examples for TOUR, which were also adopted when training re-rankers, we observed some performance variances depending on the hyperparameters. We hope to tackle this issue with better optimization in the future.

Acknowledgements

We thank Zexuan Zhong, Mengzhou Xia, Howard Yen, and the anonymous reviewers for their helpful feedback. This work was supported in part by the ICT Creative Consilience program (IITP-2023-2020-0-01819) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation), National Research Foundation of Korea (NRF-2023R1A2C3004176), and the Hyundai Motor Chung Mong-Koo Foundation.

References

Petr Baudiš and Jan Šedivỳ. 2015. Modeling of the question answering task in the yodaqa system. In *International Conference of the cross-language evaluation Forum for European languages*, pages 222–228. Springer.

Tim Baumgärtner, Leonardo F. R. Ribeiro, Nils Reimers, and Iryna Gurevych. 2022. Incorporating relevance feedback for information-seeking retrieval using few-shot document re-ranking. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8988–9005.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879.

W Bruce Croft, Donald Metzler, and Trevor Strohman. 2010. *Search engines: Information retrieval in practice*, volume 520. Addison-Wesley Reading.

Martin Fajcik, Martin Docekal, Karel Ondrej, and Pavel Smrz. 2021. R2-d2: A modular baseline for open-domain question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 854–870.

Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy J. Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Srinivasan Iyer, Sewon Min, Yashar Mehdad, and Wentau Yih. 2021. RECONSIDER: Improved re-ranking using span-focused cross-attention for open domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1280–1287.

Gautier Izacard and Edouard Grave. 2020. Distilling knowledge from reader to retriever for question answering. In *International Conference on Learning Representations*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

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

Omar Khattab, Christopher Potts, and Matei Zaharia. 2021. Relevance-guided supervision for OpenQA with ColBERT. *Transactions of the Association for Computational Linguistics*, 9:929–944.

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, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 39–48.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. 2019. Dynamic evaluation of transformer language models. *arXiv preprint arXiv:1904.08378*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Victor Lavrenko and W Bruce Croft. 2001. Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127.

Jinhyuk Lee, Mujeen Sung, Jaewoo Kang, and Danqi Chen. 2021a. Learning dense representations of phrases at scale. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6634–6647.

Jinhyuk Lee, Alexander Wettig, and Danqi Chen. 2021b. Phrase retrieval learns passage retrieval, too. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3661–3672.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096.

Patrick Lewis, Barlas Oguz, Wenhan Xiong, Fabio Petroni, Scott Yih, and Sebastian Riedel. 2022. Boosted dense retriever. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3102–3117.

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. 2021. Question and answer test-train overlap in open-domain question answering datasets. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1000–1008.

Canjia Li, Yingfei Sun, Ben He, Le Wang, Kai Hui, Andrew Yates, Le Sun, and Jungang Xu. 2018. NPRF: A neural pseudo relevance feedback framework for ad-hoc information retrieval. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4482–4491.

Hang Li, Ahmed Mourad, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. 2021. Pseudo relevance feedback with deep language models and dense retrievers: Successes and pitfalls. *Journal of ACM Transactions on Information Systems*.

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 preprint arXiv:1907.11692*.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4089–4100.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Ori Ram, Gal Shachaf, Omer Levy, Jonathan Berant, and Amir Globerson. 2022. Learning to retrieve passages without supervision. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2687–2700.

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*, pages 2825–2835.

Joseph Rocchio. 1971. Relevance feedback in information retrieval. *The Smart retrieval system-experiments in automatic document processing*, pages 313–323.

Guilherme Rosa, Luiz Bonifacio, Vitor Jeronymo, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, and Rodrigo Nogueira. 2022. In defense of cross-encoders for zero-shot retrieval. *arXiv preprint arXiv:2212.06121*.

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*, pages 6138–6148.

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-time open-domain question answering with dense-sparse phrase index. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4430–4441.

Devendra Singh, Siva Reddy, Will Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. *Advances in Neural Information Processing Systems*, 34:25968–25981.

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 *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Xiao Wang, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2021. Pseudo-relevance feedback for multiple representation dense retrieval. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 297–306.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

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*.

HongChien Yu, Chenyan Xiong, and Jamie Callan. 2021. Improving query representations for dense retrieval with pseudo relevance feedback. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3592–3596.

Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik G. Learned-Miller, and Jaap Kamps. 2018. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018*, pages 497–506.

A Derivation of the Gradient for TOUR$_{\text{hard}}$

Proof. We compute the gradient of $\mathcal{L}_{\text{hard}}(\mathbf{q}_t, \mathcal{C}_{1:k}^{q_t})$ in Eq. (10) with respect to the query representation $\mathbf{q}_t$. Denoting $\sum_{\tilde{c}} P_k(\tilde{c}|q_t)$ as Z, the gradient is:

$$
\begin{aligned}
&\frac{\partial \mathcal{L}_{\text{hard}}(\mathbf{q}_t, \mathcal{C}_{1:k}^{q_t})}{\partial \mathbf{q}_t} = \frac{\partial \mathcal{L}_{\text{hard}}(\mathbf{q}_t, \mathcal{C}_{1:k}^{q_t})}{\partial Z} \frac{\partial Z}{\partial \mathbf{q}_t} \\
&= -\frac{1}{Z} \sum_{\tilde{c}} \frac{\partial P_k(\tilde{c}|q_t)}{\partial \mathbf{q}_t} \\
&= -\frac{1}{Z} \sum_{\tilde{c}} \sum_{i=1}^{k} \frac{\partial P_k(\tilde{c}|q_t)}{\partial \mathbf{q}_t^\top \mathbf{c}_i} \frac{\partial \mathbf{q}_t^\top \mathbf{c}_i}{\partial \mathbf{q}_t} \\
&= -\frac{1}{Z} \sum_{\tilde{c}} \sum_{i=1}^{k} (\delta[c_i = \tilde{c}] - P_k(c_i|q_t)) P_k(\tilde{c}|q_t) \mathbf{c}_i \\
&= -\sum_{\tilde{c}} \big[P(\tilde{c}|q_t) \sum_{i=1}^{k} (\delta[c_i = \tilde{c}] - P_k(c_i|q_t)) \mathbf{c}_i \big] \\
&= -\sum_{\tilde{c}} P(\tilde{c}|q_t) \big[(1 - P_k(\tilde{c}|q_t)) \tilde{\mathbf{c}} - \sum_{c \in \mathcal{C}_{1:k}^{q_t}, c \neq \tilde{c}} P_k(c|q_t) \mathbf{c} \big] \\
&= -\sum_{\tilde{c}} P(\tilde{c}|q_t)(1 - P_k(\tilde{c}|q_t)) \tilde{\mathbf{c}} \\
&\quad + \sum_{\tilde{c}} \big[P(\tilde{c}|q_t) \sum_{c \in \mathcal{C}_{1:k}^{q_t}, c \neq \tilde{c}} P_k(c|q_t) \mathbf{c} \big]
\end{aligned}
$$

Then, we have:

$$
\begin{aligned}
&g(\mathbf{q}_t, \mathcal{C}_{1:k}^{q_t}) = \mathbf{q}_t - \eta \frac{\partial \mathcal{L}_{\text{hard}}(\mathbf{q}_t, \mathcal{C}_{1:k}^{q_t})}{\partial \mathbf{q}_t} \\
&= \mathbf{q}_t + \eta \sum_{\tilde{c}} P(\tilde{c}|q_t)(1 - P_k(\tilde{c}|q_t)) \tilde{\mathbf{c}} \\
&\qquad - \eta \sum_{\tilde{c}} \big[P(\tilde{c}|q_t) \sum_{c \in \mathcal{C}_{1:k}^{q_t}, c \neq \tilde{c}} P_k(c|q_t) \mathbf{c} \big].
\end{aligned}
$$

□

B Derivation of the Gradient for TOUR$_{\text{soft}}$

Proof. We compute the gradient of $\mathcal{L}_{\text{soft}}(\mathbf{q}_t, \mathcal{C}_{1:k}^{q_t})$ in Eq. (12) with respect to $\mathbf{q}_t$. Denoting $P(c_i = c^*|q_t, \phi)$ as P_i, we expand the loss term as:

$$
\begin{aligned}
&\mathcal{L}_{\text{soft}}(\mathbf{q}_t, \mathcal{C}_{1:k}^{q_t}) = -\sum_{i=1}^{k} P_i \log \frac{P_k(c_i|q_t)}{P_i} \\
&= -\sum_{i=1}^{k} P_i (\mathbf{q}_t^\top \mathbf{c}_i - \log \sum_{j=1}^{k} \exp(\mathbf{q}_t^\top \mathbf{c}_j) - \log P_i).
\end{aligned}
$$

Then, the gradient is:

$$
\begin{aligned}
&\frac{\partial \mathcal{L}_{\text{soft}}(\mathbf{q}_t, \mathcal{C}_{1:k}^{q_t})}{\partial \mathbf{q}_t} \\
&= -\sum_{i=1}^{k} P_i \frac{\partial}{\partial \mathbf{q}_t} (\mathbf{q}_t^\top \mathbf{c}_i - \log \sum_{j=1}^{k} \exp(\mathbf{q}_t^\top \mathbf{c}_j) - \log P_i) \\
&= -\sum_{i=1}^{k} P_i (\mathbf{c}_i - \frac{1}{\sum_{j=1}^{k} \exp(\mathbf{q}_t^\top \mathbf{c}_j)} \sum_{j=1}^{k} \mathbf{c}_j \exp(\mathbf{q}_t^\top \mathbf{c}_j)) \\
&= -\sum_{i=1}^{k} P_i (\mathbf{c}_i - \sum_{j=1}^{k} \mathbf{c}_j \frac{\exp(\mathbf{q}_t^\top \mathbf{c}_j)}{\sum_{l=1}^{k} \exp(\mathbf{q}_t^\top \mathbf{c}_l)}) \\
&= -\sum_{i=1}^{k} P_i (\mathbf{c}_i - \sum_{j=1}^{k} P_k(c_j|q_t) \mathbf{c}_j) \\
&= -\sum_{i=1}^{k} P_i \mathbf{c}_i + \sum_{i=1}^{k} P_k(c_i|q_t) \mathbf{c}_i.
\end{aligned}
$$

Putting it all together:

$$
\begin{aligned}
&g(\mathbf{q}_t, \mathcal{C}_{1:k}^{q_t}) = \mathbf{q}_t - \eta \frac{\partial \mathcal{L}_{\text{soft}}(\mathbf{q}_t, \mathcal{C}_{1:k}^{q_t})}{\partial \mathbf{q}_t} \\
&= \mathbf{q}_t + \eta \sum_{i=1}^{k} P(c_i|q_t, \phi) \mathbf{c}_i - \eta \sum_{i=1}^{k} P_k(c_i|q_t) \mathbf{c}_i.
\end{aligned}
$$

□

C Relation to the Rocchio Algorithm

Proof. We derive Eq. (6) from Eq. (11).

$$
\begin{aligned}
&g(\mathbf{q}_t, \mathcal{C}_{1:k}^{q_t}) \\
&= \mathbf{q}_t + \eta \sum_{\tilde{c}} P(\tilde{c}|q_t)(1 - P_k(\tilde{c}|q_t)) \tilde{\mathbf{c}} \\
&\qquad - \eta \sum_{\tilde{c}} \big[P(\tilde{c}|q_t) \sum_{c \in \mathcal{C}_{1:k}^{q_t}, c \neq \tilde{c}} P_k(c|q_t) \mathbf{c} \big] \\
&= \mathbf{q}_t + \eta \sum_{i=1}^{k'} \frac{1}{k'} (1 - \frac{1}{k}) \mathbf{c}_i \\
&\qquad - \eta \sum_{i=1}^{k'} \big[\frac{1}{k'} \sum_{j=1, i \neq j}^{k} \frac{1}{k} \mathbf{c}_j \big] \\
&= \mathbf{q}_t + \eta \frac{k-1}{k'k} \sum_{i=1}^{k'} \mathbf{c}_i \\
&\qquad - \eta \frac{1}{k'k} \sum_{i=1}^{k'} \big[\sum_{j=1, i \neq j}^{k'} \mathbf{c}_j + \sum_{j=k'+1}^{k} \mathbf{c}_j \big] \\
&= \mathbf{q}_t + \eta \frac{k-1}{k'k} \sum_{i=1}^{k'} \mathbf{c}_i \\
&\qquad - \eta \frac{1}{k'k} \big[(k'-1) \sum_{i=1}^{k'} \mathbf{c}_i + k' \sum_{i=k'+1}^{k} \mathbf{c}_i \big] \\
&= \mathbf{q}_t + \eta \frac{k-k'}{k'k} \sum_{i=1}^{k'} \mathbf{c}_i - \eta \frac{1}{k} \sum_{j=k'+1}^{k} \mathbf{c}_j.
\end{aligned}
\tag{15}
$$

Then, the equality holds when $\alpha = 1$, $\beta = \eta \frac{k-k'}{k}$, and $\gamma = \eta \frac{k-k'}{k}$. □

D Implementation Details

Phrase re-ranker To train a cross-encoder re-ranker for phrase retrieval (§3.4), we first annotate the top 100 retrieved results from DensePhrases. We use three sentences as our context, one that contains a retrieved phrase and the other two that surround it. This leads to faster inference than using the whole paragraph as input while preserving the performance. During the 20 epochs of training, we sample positive and negative contexts for every epoch while selecting the best re-reanker based on the validation accuracy of the re-ranker. We modified the code provided by the Transformers library[3] (Wolf et al., 2020) and used the same hyperparameters as specified in their documentation except for the number of training epochs. The ablation study in Table 6 shows that we can achieve stronger performance by prepending titles to inputs, using larger language models, using three sentences as our context, and pre-training over reading comprehension datasets. Using entire paragraphs as input contexts only slightly increases performance compared to using three sentences, but it doubles the query latencies of re-ranking.

	NQ
Phrase re-ranker	45.4
Without prepending titles	44.8
Rl ⇒ Rb	43.2
3 ⇒ 1 sentence	43.6
3 ⇒ Paragraph*	**45.6**
RC ⇒ MNLI pre-training	43.8
RC ⇒ No pre-training	42.0

Table 6: Ablation study of our phrase re-ranker. Rl: RoBERTa-large. Rb: RoBERTa-base. RC: reading comprehension. *: using entire paragraphs as input doubles query latencies.

Dense retriever We modified the official code of DensePhrases[4] (Lee et al., 2021a) and DPR[5] (Karpukhin et al., 2020) to implement TOUR on dense retrievers. While pre-trained models and indexes of $\text{DensePhrases}_{\text{multi}}$ and DPR_{NQ} are publicly available, the indexes of $\text{DensePhrases}_{\text{NQ}}$ and $\text{DPR}_{\text{multi}}$ have not been released as of May 25th, 2022. When necessary, we reimplemented them to experiment with open-domain QA and passage retrieval in the query distribution shift setting.

Hyperparamter When running TOUR , we use gradient descent with momentum set to 0.99 and use weight decay $\lambda_{\text{decay}} = 0.01$. We also perform a linear learning rate scheduling per iteration. Both the threshold p and temperature τ for pseudo labels are set to 0.5. Table 7 lists the hyperparameters that are used differently for each task. All hyperparameters of TOUR were tuned using the in-domain development set.

	ODQA	Passage Retrieval	
Hyperparameter	DensePhrases	DensePhrases	DPR
Learning rate η	1.2	1.2	0.2
Max iterations	3	1	1
Retrieval top-k	10	100	100
Re-ranker top-k	10	100	100
Re-ranker λ	0.1	1	1

Table 7: Hyperparameters of TOURfor open-domain QA (ODQA) and passage retrieval.

E Data Statistics

Dataset	**Train**	**Dev**	**Test**
Natural Questions	79,168	8,757	3,610
TriviaQA	78,785	8,837	11,313
WebQuestions	3,417	361	2,032
CuratedTrec	1,353	133	694
SQuAD	78,713	8,886	10,570
EntityQuestions	-	-	22,075

Table 8: Statistics of open-domain QA datasets.

Table 8 shows the statistics of the datasets used for end-to-end open-domain QA and passage retrieval tasks. For EntityQuestions, we only use its test set for the query distribution shift evaluation.

[3] https://github.com/huggingface/transformers/blob/v4.13.0/examples/pytorch/text-classification/run_glue.py

[4] https://github.com/princeton-nlp/DensePhrases

[5] https://github.com/facebookresearch/DPR

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Yes, please see the limitations section.

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Yes, please see the abstract and introduction sections.

☑ A4. Have you used AI writing assistants when working on this paper?
Grammarly for grammar checking. The contents are original work from the human authors.

B ☑ Did you use or create scientific artifacts?

Yes, please see section 4.1 Datasets.

☑ B1. Did you cite the creators of artifacts you used?
Yes, please see section 4.1 Datasets.

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Yes, please see section 4.1 Datasets.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Yes, please see section 4.1 Datasets.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Yes, please see section 4.1 Datasets.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Yes, please see Appendix E Data Statistics.

C ☑ Did you run computational experiments?

Yes, please see section 4 Experiments.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Yes, please see section 4 Experiments.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Yes, please see appendix F Implementation Details.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Yes, please see section 4 Experiments.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Yes, please see appendix F Implementation Details.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.