Multifaceted Improvements for Conversational Open-Domain Question Answering

Tingting Liang
Hangzhou Dianzi University
Hangzhou, China
liangtt@hdu.edu.cn

Yixuan Jiang
Hangzhou Dianzi University
Hangzhou, China
jyx201050027@hdu.edu.cn

Congying Xia
University of Illinois at Chicago
Chicago, US
cxia@uic.edu

Ziqiang Zhao
Hangzhou Dianzi University
Hangzhou, China
zhaoziqiang@hdu.edu.cn

Yuyu Yin
Hangzhou Dianzi University
Hangzhou, China
yinyuyu@hdu.edu.cn

Philip S. Yu
University of Illinois at Chicago
Chicago, US
psyu@uic.edu

ABSTRACT
Open-domain question answering (OpenQA) is an important branch of textual QA which discovers answers for the given questions based on a large number of unstructured documents. Effectively mining correct answers from the open-domain sources still has a fair way to go. Existing OpenQA systems might suffer from the issues of question complexity and ambiguity, as well as insufficient background knowledge. Recently, conversational OpenQA is proposed to address these issues with the abundant contextual information in the conversation. Promising as it might be, there exist several fundamental limitations including the inaccurate question understanding, the coarse ranking for passage selection, and the inconsistent usage of golden passage in the training and inference phases. To alleviate these limitations, in this paper, we propose a framework with Multifaceted Improvements for Conversational open-domain Question Answering (MICQA). Specifically, MICQA has three significant advantages. First, the proposed KL-divergence based regularization is able to lead to a better question understanding for retrieval and answer reading. Second, the added post-ranker module can push more relevant passages to the top placements and be selected for reader with a two-aspect constrains. Third, the well designed curriculum learning strategy effectively narrows the gap between the golden passage settings of training and inference, and encourages the reader to find true answer without the golden passage assistance. Extensive experiments conducted on the publicly available dataset OR-QuAC demonstrate the superiority of MICQA over the state-of-the-art model in conversational OpenQA task.

CCS CONCEPTS
• Computer systems organization → Embedded systems; Redundancy; Robotics; • Networks → Network reliability.

KEYWORDS
Open Domain, Conversational Question Answering, Curriculum Learning

ACM Reference Format:
Tingting Liang, Yixuan Jiang, Congying Xia, Ziqiang Zhao, Yuyu Yin, and Philip S. Yu. 2018. Multifaceted Improvements for Conversational Open-Domain Question Answering. In Woodstock ’18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION
Open-domain question answering (OpenQA) aims to discover answers from an extremely large text source such as Wikipedia for given questions [3, 32, 34]. OpenQA that receives the more widespread application as its setting is more aligned with real-world QA process of human beings. Generally, an OpenQA system needs to firstly locate a small collection of relevant articles and then generate the answer. The widely adopted architecture of the existing OpenQA system consists of two components, retriever and reader. The former acts as an information retrieval system to find relevant passages probably with the correct answer contained [9, 13, 15, 19, 31]. The latter aims at extracting or generating the answers from the retrieved passages. Although OpenQA enjoys a sound development momentum, it is challenged by several issues including question complexity and ambiguity, as well as insufficient background knowledge. Conversational OpenQA which executes question answering with open-domain sources under the conversational setting is able to address the above issues by providing the context information in the conversations.

Previous work [21] firstly defines the task of open-retrieval conversational question answering (ORConvQA) and proposes an effective end-to-end system to deal with it. Promising as it might be, several fundamental limitations still exist: (1) Inaccurate question understanding. The questions in the dialogues might be hard to understand due to the unspecified pronouns. To alleviate this issue, ORConvQA uses rewrite questions by replacing pronouns with real
entities to pre-train the retriever. Then, they concatenate the dialogue history and the question together to jointly train the whole model. However, the input of the pre-train and the joint training process is still often inconsistent which makes the model incapable of really understand those unclear questions. (2) Coarse ranking.

Previous works adopt the retriever-reader pipeline which consists of a passage retriever and answer reader with a reranker/selector as regularization [11, 21, 32]. The reranker/selector here does not influence the retriever and the reader only takes the top-ranked passages as the input. This makes the whole pipeline suffer from the coarse ranking, especially for the situations when golden passages are not retrieved in the top-ranked passages. (3) Inconsistent usage of golden passage. During training, ORConvQA adds the golden passage into the training manually when the golden passage is not retrieved in the top-ranked passages. This is impractical for the testing as the golden passages are unknown in the inference stage. The inconsistency usages of golden passage during training and testing might lead to poor performance when the golden passage cannot be accurately retrieved.

To address the previous issues, we propose a framework which has Multifaceted Improvements for Conversational open-domain Question Answering (MICQA). MICQA improves the previous work from multiple aspects including regularizing retriever pre-training, as well as incorporating post-ranking and curriculum learning. First, Figure 1(a) shows that, a Kullback-Leibner (KL)-divergence based regularization is proposed for retriever pre-training, which constrains the retrieved results outputted by feeding different forms of questions. In this manner, we can learn a better representation for the concatenated question by taking its rewrite as supervision and minimizing the information loss between them. The better question understanding would benefit the retrieval and question answering performance in different phases. Second, to further improve the initial retrieval results, we apply a multi-stage pipeline by adding a post-ranker module as shown in Figure 1(b), which also known as the post-processing [34] over retrieved passages generated by the retriever. The post-ranker learning benefits from a two-aspect constrain posed by a ranking loss with a distance-based contrastive loss. In this way, the relevant passages would be enforced by post-ranker to appear at the top positions and selected for answer reading. Third, in the joint training stage, we design a semi-automatic curriculum learning strategy (the blue dashed boxes in Figure 1(c)) to reduce the dependency on the golden passages. It encourages the model to find true answer spans even when the golden passage are not retrieved as the top-ranked passages. We evaluate MICQA on the publicly released dataset OR-QuAC [21] to demonstrate its effectiveness.

Our main contributions are as follows:

- We propose a KL-divergence based regularization for retriever pre-training, which greatly strengthens the capability of question understanding.
- We propose a multi-stage pipeline for the conversational OpenQA task by incorporating a post-ranker module in the conventional retriever-reader pipeline.
- We design a curriculum learning strategy for the joint training of the retriever, post-ranker, and reader, which effectively closes the gap between training and inference.
- We conduct extensive experiments on the OR-QuAC dataset to demonstrate the effectiveness of the proposed MICQA. We further provide in-depth analysis on model ablation and case study to explore the insights of MICQA.

2 PRELIMINARY

2.1 Retriever-Reader Pipeline

The most typical OpenQA system follows a two-stage pipeline, which has two components: one retriever and one reader.

2.1.1 Retriever. In the retrieval process, the commonly used dual-encoder model [2] consists of a question encoder and a passage encoder, which encode the question and passage into low-dimensional vectors, respectively. Many similarity functions, such as inner product and Euclidean distance, can be used to measure the relationship between questions and passages. The evaluation in [11] demonstrates that these functions perform comparably, so the simpler inner product is selected in our work:

\[
\text{sim}(q,p) = E_Q(q)^T E_P(p).
\]

where \( q \) and \( p \) denote a given question and passage. \( E_Q \) and \( E_P \) refer to the question encoder and passage encoder which typically are two pre-trained model, e.g., BERT[5], ALBERT[14]. The retriever score is defined as the similarity of representations of the question and passage. Given a question \( q \), the retriever derives a small subset of passages with embeddings closest to \( q \) from a large corpus.
The overall framework of our MICQA is shown in Figure 2. MICQA consists of three components: (1) one regularized retriever for relevant passages discovery, (2) one contrastive post-ranker for improving passage retrieval quality, and (3) one reader for detecting answers from a small collection of ranked passages. These three components work as follows: With the passage representations offline encoded by the pre-trained passage encoder, the retriever outputs the top K passages by operating inner product between them and the embedding of a given question generated by the question encoder. The post-ranker takes the initially retrieved top K passage embeddings and question embedding as input, and selects the top T of the newly ranked passages as output. The outputted passages are respectively concatenated with the given question, and the concatenations are fed into the reader to find the final answer. The answer score is decided by the post-ranker score and reader score, which would be introduced in details in Section 3.3. The retriever is pre-trained with a KL-divergence based regularization. Then the passage encoder of it is frozen and the question encoder can be fine-tuned with the other two components in the next joint training process in an end-to-end manner. In the following part, we would describe the procedures of pre-training, joint training, and inference, along with the details of each component.

### 3.1 KL-Divergence based Regularization for Retriever Pre-training

In order to improve the capability of question understanding, we propose to exploit both the original questions and question rewrites in retriever pre-training. Specifically, a regularization mechanism is proposed to force two distributions of the retrieved results generated by feeding the original questions and their rewrites into the dual-encoder to be consistent with each other by minimizing the bidirectional KL divergence between them.

Concretely, given the current question \(q_i\) and its historical question-answer pairs \(\{q_{i}, a_{i}\}_{i=1}^{c} \), we form the original question by concatenating the question-answer pairs in a history window of size \(w\) with the current question. Moreover, to mitigate the issue of under-specified and ambiguous initial questions, the initial question \(q_i\) is invariably considered as it makes the constructed original question closer to the question rewrite. For the question encoder, the input sequence of original question can be defined as \(q_{i}^{or} = [CLS] q_{i} \ [SEP] a_{1} \ [SEP] q_{i-w} \ [SEP] a_{c-w} \ [SEP] \ldots \ [SEP] a_{c-1} \ [SEP] q_{c} \ [SEP] \).

Let \(\mathcal{D} = \{(q_{i}^{or}, q_{i}^{rw}, p_{1}^{c}, p_{2}^{c}, \ldots, p_{m}^{c})_{i=1}^{m}\}\) denote the training data that consists of \(m\) instances. Each instance contains two forms of one question (i.e., original question and question rewrite), one positive passage, along with \(n\) irrelevant passages \(\{p_{j}^{c}\}_{j=1}^{n}\). These irrelevant passages used for training contain one hard negative of the given question and \(n-1\) in-batch negatives which are the positive and negative samples of the other questions from the same mini-batch. As shown in Figure 3, we feed the original question \(q_{i}^{or}\)
We can obtain two distributions of the retrieved results through pre-training our MICQA jointly trains the question encoder, post-ranker, and reader with a designed curriculum learning strategy.

3.2 Joint Training with Curriculum Learning

With the passage encoder and question encoder of the retriever pre-trained, our MICQA jointly trains the question encoder, post-ranker, and reader with a designed curriculum learning strategy.

3.2.1 Retriever Loss. The pre-trained passage encoder is used to embed the open-domain passages offline and obtain a set of passage representations. Given the current question \( q_i \), and its historical question-answer pairs \((q_1, a_1), \ldots, (q_{i-c+1}, a_{i-c+1})\) within a window size \( w \), the input for question encoder is constructed by concatenating the question-answer pairs with the current question as mentioned in Section 3.1. With the prepared offline passage representations and outputs of question encoder, each passage would be assigned with a retrieval score computed by the inner product operation as shown in Equation 1. We select the top \( K \) passages with high retrieval scores for the post-ranker and reader.

For each question, the reformatted question for the question encoder is denoted as \( q_i^\text{en} \). The question encoder of the retriever is fine-tuned by optimizing the following retrieval loss:

\[
L_{\text{retriever}}^i = - \log \frac{\exp(\text{sim}(q_i^\text{en}, p_i^+))}{\sum_{j \neq i} \exp(\text{sim}(q_i^\text{en}, p_i^-))},
\]

where \( p_{i,j} \) denotes the retrieved passages regarding question \( q_i \).

For a fair comparison, we omit the answers from the concatenated question in the experiment to keep the same setting with the previous work [21].

3.2.2 Post-Ranker with Contrastive Loss. Since only the first \( T \) of the top \( K \) passages outputted by the retriever module are fed into the reader for answer extraction, a post-ranker is proposed to re-rank the \( K \) passages to push more relevant passages be ranked in the top \( T \) placement. We build the post-ranker by adding a subsequent network after the pre-trained passage encoder, which takes the embeddings of \( K \) passages as input. With the subsequent network, post-ranker is able to learn high-level feature representations for passages. In this work, we use the linear layer as the subsequent network for simplicity. The output for the retrieved passage \( p_{i,j} \) and the ranking score can be formulated as follows:

\[
d_{i,j} = \text{LinearLayer}(E_P(p_{i,j})),
\]

\[
S_{\text{post}}(q_i^\text{en}, p_{i,j}) = q_i^\text{T}d_{i,j},
\]

where \( d_{q_i} \) is the representation for question \( q_i \) generated by the question encoder \( E_Q \). \( S_{\text{post}}(\cdot) \) refers to the scores generated by the post-ranker network.

To simultaneously fine-tune the question encoder and post-ranker, we apply the modified hinge loss combined with a distance-based contrastive loss to pose constraints for passage reranking from two aspects. The hinge loss function [26] is defined as:

\[
L_{\text{ranker}} = \max \{ 0, \delta - S_{\text{post}}(q_i^\text{en}, p_i^+) + \max_j (S_{\text{post}}(q_i^\text{en}, p_{i,j})) \},
\]

where \( \delta \) is the margin of the hinge loss. \( p_i^+ \) and \( p_{i,j} \) denote the positive passage and negative passages retrieved for \( q_i \). For contrastive learning, we use the triplet margin loss [29] to measure the distance between positive and negative samples as follows:

\[
L_{\text{tr}} = \max \{ 0, \mu + D(d_{q_i}, d_i^+) - D(d_{q_i}, d_i^-) \},
\]

where \( \mu \) is the margin of the triplet margin loss. We apply Euclidean distance \( D \) to compute the distance between the question and question rewrite \( q_i^\text{en} \) to the question encoder \( E_Q \), respectively. The derived question embeddings are matched with the passage embeddings outputted by the passage encoder \( E_P \) via inner product. We can obtain two distributions of the retrieved results through softmax operation, denoted as \( p_{\text{or}}(p_i|q_i^\text{en}) \) and \( p_{\text{rw}}(p_i|q_i^\text{en}) \). In the retrieval pre-training phase, we try to regularize on the retrieved results by minimizing the bidirectional KL divergence between the two distributions, which can be formulated as follows:

\[
L_{KL} = \frac{1}{2} \left( D_{KL}(p_{\text{or}}(p_i|q_i^\text{en}) || p_{\text{rw}}(p_i|q_i^\text{en})) + D_{KL}(p_{\text{rw}}(p_i|q_i^\text{en}) || p_{\text{or}}(p_i|q_i^\text{en})) \right).
\]

For the main task of passage retrieval, we apply the widely used negative log likelihood of the positive passage given two forms of one question as objective function:

\[
L_{\text{NLL}} = \frac{1}{2} (L_{\text{NLL, or}} + L_{\text{NLL, rw}}),
\]

\[
= - \frac{1}{2} (\log p_{\text{or}}(p_i^+|q_i^\text{en}) + \log p_{\text{rw}}(p_i^+|q_i^\text{en})).
\]

The probability of retrieving the positive passage can be calculated as:

\[
p_{\text{or}}(p_i^+|q_i^\text{en}) = \frac{\exp(\text{sim}(q_i^\text{en}, p_i^+))}{\sum_{j \neq i} \exp(\text{sim}(q_i^\text{en}, p_i^-))}.
\]

\[
p_{\text{rw}}(p_i^+|q_i^\text{en}) = \frac{\exp(\text{sim}(q_i^\text{en}, p_i^+))}{\sum_{j \neq i} \exp(\text{sim}(q_i^\text{en}, p_i^-))}.
\]

The probability of retrieving those negative passages can be obtained in the same manner.

To pre-train the retriever, the training objective is to minimize the pre-retrieval loss \( L_{\text{pre}} \) for data \((q_i^\text{en}, q_i^\text{rw}, p_i^+, p_{i,1}, \ldots, p_{i,n})\):

\[
L_{\text{pre}} = \frac{1}{2} L_{\text{NLL}} + \alpha L_{KL}.
\]

where \( \alpha \in [0, 1] \) is a hyperparameter used to control \( L_{KL} \).
and passage in the representation space. The final post-ranker loss is defined as:
\[ \mathcal{L}^{i}_{\text{postranker}} = \mathcal{L}^{i}_{\text{ranker}} + \beta \mathcal{L}^{i}_{\text{cl}}, \quad (12) \]
where \( \beta \) is hyperparameter that need to be determined.

3.2.3 Reader Loss. The neutral reader of our MICQA is designed for predicting an accurate answer to the given question. Given the top \( T \) retrieved/re-ranked passages, the reader intends to detect an answer span from each passage with a span score. Moreover, a passage selection score is assigned to each passage. The passage selection is used to select the passages that contains the true answer of the given question [11, 17], which performs as a constraint to facilitate the training of reader. The span with the highest passage selection score and span score is extracted as the final answer.

Our reader applies the widely used pre-trained model, such as BERT [5], RoBERTa [18]. Given a question \( q_1 \) and the top \( T \) previously retrieved passages, we form the question input in the same way as that for retriever described in Section 3.2.1. The only difference is that the initial question \( q_1 \) is left out as the setting of [21]. We denote the reconstructed question as \( q^d_1 \) and concatenate it with a retrieved passage to form the sequence as the input for the pre-trained model. Specifically, let \( z_{i,j} \) be the outputted token-level representations of the \( t \)-th token in the passage \( p_{i,j} \) which is retrieved for question \( q^d_1 \). \( z_{i,j,[\text{cls}]} \) denotes the sequence-level representation for the input sequence. The scores for the \( t \)-th token being the start and end tokens, and a passage being selected are defined as follows:
\[ S_1(q^d_1, p_{i,j}, |t|) = z_{i,j,[\text{cls}]}, \quad (13) \]
\[ S_2(q^d_1, p_{i,j}, |t|) = z_{i,j+[\text{cls}]}, \quad (14) \]
\[ S_{\text{select}}(q^d_1, p_{i,j}) = z_{i,j,[\text{cls}]}, \quad (15) \]
where \( w_{\text{start}} \), \( w_{\text{end}} \), and \( w_{\text{select}} \) are trainable vectors.

The loss function for the start token prediction is defined as:
\[ \mathcal{L}^{i}_{\text{start}} = -\log \frac{\exp(S_1(q^d_1, p_{i,j}, |t|))}{\sum_{j=1}^{M} \exp(S_1(q^d_1, p_{i,j}, |t|))}, \quad (16) \]
where \( [s^*_t] \) denotes the start token of the true answer in the golden passage \( p^*_t \). The objective function for the end token prediction denoted as \( \mathcal{L}^{i}_{\text{end}} \) is defined in the same manner. In addition, the passage selection loss is computed as follows:
\[ \mathcal{L}^{i}_{\text{select}} = -\log \frac{\exp(S_{\text{select}}(q^d_1, p_{i,j}))}{\sum_{j=1}^{M} \exp(S_{\text{select}}(q^d_1, p_{i,j}))}. \quad (17) \]
Finally, the loss function of the reader is defined as follows:
\[ \mathcal{L}^{i}_{\text{reader}} = \frac{1}{2} \mathcal{L}^{i}_{\text{start}} + \mathcal{L}^{i}_{\text{end}} + \mathcal{L}^{i}_{\text{select}}. \quad (18) \]

3.2.4 Joint Training with Curriculum Learning Strategy. We propose a semi-automatic curriculum learning (CL) strategy consisting of two core components: an automatic difficulty measurer and a discrete training scheduler, to improve the joint training of retriever, post-ranker, and reader. The CL strategy aims at reducing the chance of adding golden passage during the joint training process to help the settings of training and inference be more consistent. It would effectively push the retriever to find the passage with the true answer contained by itself and encourage the reader to discover the correct answer without the assistance of golden passage. To the best of our knowledge, this is the first work that designs CL strategy for conversational OpenQA joint learning.

Specifically, our semi-automatic CL strategy automates the difficulty measurer by taking the question-wise training loss as criteria. A higher retriever loss is recognized a harder mode for the retriever to discover the positive passages. For the hard mode, the training scheduler of our CL strategy introduces the golden passage in the retrieval results to make the learning easier. Our training scheduler is like a Baby Step scheduler [1, 27] with a finer granularity as shown in Algorithm 1.

Formally, for each iteration \( l \), let \( \mathcal{P}^{(l)}_K = \{ \mathcal{P}^{(l)}_i | \mathcal{Q}^{(l)}_{i} \in \mathcal{Q}^{(l)} \} \) be the set of all the collections of top \( K \) passages retrieved by the retriever to questions of the batch in the iteration. For the hard mode, the training scheduler brings the golden passage of each question to \( \mathcal{P}^{(l)}_{K_G} \) to form the set of retrieved passages with the positive one, denoted as \( \mathcal{P}^{(l)}_{KG} = \{ \mathcal{P}^{(l)}_{i} | \mathcal{Q}^{(l)}_{i} \in \mathcal{Q}^{(l)} \} \). With these retrieved passages, the final loss to be optimized is defined as:
\[ \mathcal{L}^{(l)}_{\text{final}} = \mathcal{L}^{(l)}(\mathcal{P}^{(l)}_{KG}) + (1 - \sigma^{(l)}) \mathcal{L}^{(l)}(\mathcal{P}^{(l)}_{K} \setminus \mathcal{P}^{(l)}_{KG}), \quad (19) \]
where \( \sigma^{(l)} \) is determined by the difficulty measurer with the retriever loss of the previous iteration:
\[ \sigma^{(l)} = \begin{cases} 1, & \mathcal{L}^{(l-1)}_{\text{retriever}} > \lambda_{\text{upper}}, \\ 0, & \mathcal{L}^{(l-1)}_{\text{retriever}} < \lambda_{\text{lower}}, \\ \ell(p_b), & \text{otherwise}, \end{cases} \quad (20) \]
where \( \lambda_{\text{upper}} \) and \( \lambda_{\text{lower}} \) are the pre-defined upper and lower thresholds. \( \ell(p_b) \) denotes an indicator function whose output is sampled from a Bernoulli distribution:
\[ \ell(p_b) = \begin{cases} 1, & \text{with probability } p_b, \\ 0, & \text{with probability } 1 - p_b. \end{cases} \quad (21) \]
where the probability is evaluated by the min-max normalization denoted as \( p_b = \min_{0 \leq \lambda \leq \lambda_{\text{upper}} - \lambda_{\text{lower}}} \mathcal{L}^{(l)}_{\text{retriever}} \), which measures the degree of approximating upper threshold for the retriever loss. It can be intuitively explained that if the retrieval loss of the previous iteration is close to an upper threshold \( \lambda_{\text{upper}} \), the training status is regarded in the hard mode. And the value of \( \sigma^{(l)} \) is assigned with 1 in a higher probability to select the set of retrieved passages with golden passage for training in the current iteration.

\[ \mathcal{L}^{(l)}(\cdot) \] in Equation 19 denotes the objective function integrated by all the three modules with the retrieved passages in the \( l \)-th iteration as follows:
\[ \mathcal{L}^{(l)} = \mathcal{L}^{(l)}_{\text{retriever}} + \mathcal{L}^{(l)}_{\text{postranker}} + \mathcal{L}^{(l)}_{\text{reader}}. \quad (22) \]
The loss of each module is averaged over the questions in the \( l \)-th iteration.

3.3 Inference
In the inference stage, for a given question \( q_c \) and its historical question-answer pairs \( \{ q_{c,i}, a_{c,i} \}_{i=1}^{c-1} \), within a window size \( \omega \), we first obtain a collection of the top \( T \) relevant passages through the two consecutive modules, namely retriever and post-ranker. For each passage \( p_{c,j} \) in the collection, we get the post-ranker score \( S^{post}(q^*_c, p_{c,j}) \) according to Equation 9. The reader score includes
where To evaluate the effectiveness of the proposed MICQA, we conduct
Compute the coefficient 11
Optimize L
for the evaluation of the retrieval performance. MRR reflects the
is true for every questions in the dialog (HEQ-D). Furthermore, we
examples for which system F1 exceeds or matches human F1. Here
good as that of an average human. It measures the percentage of
stopwords. HEQ is used to judge whether a system’s output is as
pretraining a better passage retriever. OR-QuAC also provides a
collection of more than 11 million passages obtained from the English
Wikipedia dump from 10/20/20191 for open-retrieval. Table 1 summarizes the statistics of the aggregated OR-QuAC dataset.

4.2 Experimental Settings

4.2.1 Evaluation Metrics. Following the evaluation protocols used in [21], we apply the word-level F1 and the human equivalence score (HEQ) that are provided by the QuAC challenge [4] to evaluate our MICQA. As the core evaluation metric for the overall performance of answer retrieval, F1 is computed by considering the portion of words in the prediction and ground truth that overlap after removing stopwords. HEQ is used to judge whether a system’s output is as good as that of an average human. It measures the percentage of examples for which system F1 exceeds or matches human F1. Here two variants are considered: the percentage of questions for which this is true (HEQ-Q), and the percentage of dialogs for which this is true for every questions in the dialog (HEQ-D). Furthermore, we use another two metrics, Mean Reciprocal Rank (MRR) and Recall for the evaluation of the retrieval performance. MRR reflects the abilities of post-ranker to return the passages containing true answers in a high place. Recall is indicative of post-ranker’s capability of providing relevant passages for the next modules. For the sake of fairness, we follow [21] and calculate the two metrics for the top T passages that are retrieved for the reader.

4.2.2 Implementation Settings.

(1) Retriever Pretraining. As question and passage encoders, we employ two ALBERT Base models. The maximum sequence length for the question encoder is 128, while the maximum length for the passage encoder is 384. The models are trained on NVIDIA TITAN X GPU. The training batch size is set to 16, the number of training epochs is set to 12, the learning rate is set to 5e-5, the window size w is set to 6 and the coefficient of KL divergence α is set to 0.2. Every 5,000 steps, we save checkpoints and assess on the development set provided by [21]. Then we select several models that performed well on the development set and apply them on test questions and wiki passages. Finally, we select the best model for future training.

(2) Post-ranker and Reader. For the post-ranker, we utilize a linear layer with a size of 128 for simplicity. The number K of passage embeddings it takes as input is set to 100, and the number T of its outputted passages for reader is set to 5.

For the reader, we apply the BERT and RoBERTa model. The

### Table 1: Data statistics of the OR-QuAC dataset.

| Items                          | Train | Dev  | Test |
|-------------------------------|-------|------|------|
| # Dialogs                     | 4,383 | 490  | 771  |
| # Questions / Rewrites        | 31,526| 3430 | 5571 |
| # Avg. Questions / Dialog     | 7.2   | 7    | 7.2  |
| # Avg. Tokens / Question      | 6.7   | 6.6  | 6.7  |
| # Avg. Tokens / Rewrite       | 10    | 10   | 9.8  |
| Wikipedia                     |       |      | 11 million |

1 https://dumps.wikimedia.org/enwiki/20191020/

### Algorithm 1: Training Scheduler

```
Input: Training data \( D_{train} = \{ (q_i, a_i, p_i^*) \}_{i=1}^m \), Iteration number \( L \).
Output: A set of optimal model parameters.

for \( l = 1, \ldots, L \) do
    Sample a batch of questions \( Q^{(l)} \)
    for each question \( q_i \in Q^{(l)} \) do
        \( \mathcal{P}_i^{(l)} \leftarrow \arg\max_{p_{ij}} (\text{sim}(q_i^n, p_{ij}), K) \)
        \( \mathcal{P}_i^{(l)} \leftarrow \mathcal{P}_i^{(l)} \cup \{ p_i^+ \} \)
        Compute loss terms \( \mathcal{L}_{\text{retriever}}^{(l)}, \mathcal{L}_{\text{postraker}}^{(l)}, \mathcal{L}_{\text{reader}}^{(l)} \)
        according to Eq.7, Eq.10, Eq.18
    end
    \( \mathcal{L}^{(l)} \leftarrow \frac{1}{|Q^{(l)}|} \sum_l (\mathcal{L}_{\text{retriever}}^{(l)} + \mathcal{L}_{\text{postraker}}^{(l)} + \mathcal{L}_{\text{reader}}^{(l)}) \)
    \( \mathcal{P}_j^{(l)} \leftarrow \{ p_i^j \mid q_i \in \mathcal{Q}^{(l)} \}, \mathcal{P}_j^{(l)} \leftarrow \{ \mathcal{P}_j^{(*)} \mid q_i \in \mathcal{Q}^{(l)} \} \)
    Compute the coefficient \( v^{(l)} \) according to Eq. 20
    if \( v^{(l)} = 1 \) then
        \( \mathcal{L}_{\text{final}}^{(l)} \leftarrow \mathcal{L}^{(l)} (\mathcal{P}_j^{(*)}) \)
    else
        \( \mathcal{L}_{\text{final}}^{(l)} \leftarrow \mathcal{L}^{(l)} (\mathcal{P}_j^{(*)}) \)
    end
    Optimize \( \mathcal{L}_{\text{final}}^{(l)} \)
end
```
### Table 2: Performance comparison of MICQA and baseline models. The number in the parentheses is the batch size during the retriever pre-training.

| Methods            | Development | Test |
|--------------------|-------------|------|
|                    | F1 | HEQ-Q | HEQ-D | Rt MRR | Rt Recall | F1 | HEQ-Q | HEQ-D | Rt MRR | Rt Recall |
| DrQA [3]           | 4.5| 0.0   | 0.0   | 0.1151 | 0.2000    | 6.3| 0.1   | 0.0   | 0.1574 | 0.2253    |
| BERTserini [32]    | 19.3| 14.1  | 0.2   | 0.1767 | 0.2656    | 26.0| 20.4  | 0.1   | 0.1784 | 0.2507    |
| DPR (16) [11]      | 25.9| 16.4  | 0.2   | 0.3993 | 0.5440    | 26.4| 21.3  | 0.5   | 0.1739 | 0.2447    |
| ORConvQA-bert (64) [21] | 26.9| 17.5  | 0.2   | 0.4286 | 0.5714    | 29.4| 24.1  | 0.6   | 0.2246 | 0.3141    |
| ORConvQA-roberta (64) | 26.5| 17.8  | 0.2   | 0.4284 | 0.5624    | 28.7| 24.2  | 0.8   | 0.2330 | 0.3226    |
| ours-bert (16)     | 28.0| 19.4  | 0.2   | 0.4639 | 0.6157    | 31.7| 27.8  | 1.2   | 0.2763 | 0.3668    |
| ours-roberta (16)  | 28.1| 19.5  | 0.4   | 0.4639 | 0.6169    | 33.4| 29.4  | 1.7   | 0.2887 | 0.3819    |
| ours-bert (32)     | 27.6| 19.6  | 0.0   | 0.4675 | 0.6236    | 32.6| 29.1  | 0.8   | 0.3013 | 0.4130    |
| ours-roberta (32)  | 29.4| 20.2  | 0.4   | 0.4656 | 0.6248    | 35.0| 30.8  | 1.8   | 0.3073 | 0.4202    |

- **DrQA [3]** is composed of a document retriever which uses bigram hashing and TF-IDF matching to return the relevant passages for a given question, and a multi-layer RNN based document reader for answer spans detection in those retrieved passages.
- **BERTserini** [32] uses a BM25 retriever from Anserini\(^2\) and a BERT reader to tackle end-to-end question answering. The retriever directly identifies segments of open-domain texts and pass them to the reader. Compared to DPR, ORConvQA and our MICQA, it has no selection loss in the reader and benefits less from the joint learning.
- **DPR** [11] increases retrieval by learning dense representations instead of using typical IR methods. It innovatively proposes to introduce hard negatives in the training process of the retriever. For OpenQA, DPR consists of a dual encoder as a retriever and BERT as a reader.
- **ORConvQA** [21] is first proposed to solve the conversational open-domain QA problem with the retriever, rerank, and reader pipeline. This is the key work to compare for our MICQA.

The results of all the baselines except DPR come from the previous work [21]. The implementation setting of DPR is the same with ORConvQA, including the encoder network, window size, learning rate, and so on.

### 4.3 Overall Results

The overall experimental results are reported in Table 2. The results of the baseline models are public in [21] except for DPR. The retrieval metrics denoted by “Rt MRR”, and “Rt Recall” are used to evaluate the retrieval results of retriever for baselines while evaluating that of post-ranker for MICQA. Generally, our MICQA outperforms all the baseline models. In detail, several observations can be achieved:

1. **Both DrQA and BERTserini achieve poor performance.** The primary reason is they use the sparse retriever which cannot be fine-tuned in the downstream reader training to discover relevant passages for answer extraction. DrQA performs rather badly in answer reading as it uses RNN-based reader which does not give the strong ability of representation learning as those pre-trained language model. The reader of BERTserini is similar to the other compared methods except DrQA. But it benefits less from the multi-task training process as there is no select component (reranker) in reader. Compared with DrQA and BERTserini, DPR improves the retriever with the dense representation learning. The performance of DPR is limited by the batch size of pre-training.

2. **As the first system designed for the task of conversational OpenQA, ORConvQA provides the best performance among the baselines.** ORConvQA is similar with DPR, where the difference is that it does not use hard negatives for retriever pre-training while DPR uses. The main reason ORConvQA performs better is the batch size of retriever pre-training is

---

\(^2\)http://anserini.io/
4.4 Ablation Studies

To investigate the effectiveness of three improved parts in our MICQA, we evaluate some variants of our system. As shown in Table 3, once we remove one of the three components, both the retrieval and QA performance generally decrease. The detailed observations are summarized as follows:

(1) When we remove the KL-based regularized pre-trained retriever and use the retriever of ORConvQA as replacement, the performance drops significantly, especially the retrieval performance. It shows the importance of the KL-based regularization in our pre-trained retriever.

(2) Removing the post-ranker also brings a degradation in the overall performance. The influence is slightly smaller than that of KL-based regularization, which is probably caused by the simple linear layer used for post-ranger. This is our limitation and we plan to explore the more flexible and effective neural network for post-ranker in future work. From another perspective, some improvements can be achieved just by adding a linear layer for further passage representation learning. It is notable that the retrieval recall is higher than the full system. It is mainly because each question in the test set has more than one golden passage. Retrieving more golden passages does not necessarily lead to the better answer span which has the higher coverage of the ground truth answer.

(3) The variant without curriculum learning in the joint training performs worse than the full MICQA. By comparison, the QA performance decreases more noticeably. The reason behind is that the curriculum learning strategy encourages the reader to find correct answer with no golden passage assistance at the joint training time. It makes the reader more suitable for answer extraction in the inference phase.

4.5 Further Analysis

4.5.1 Retriever Performance. We evaluate some existing pre-trained retriever with our KL-divergence regularized retriever on the full collection of Wikipedia passages. The evaluation results are reported in Table 4. As the typical sparse retriever, BM25 achieves a good retrieval performance while its QA performance is limited as it can not be fine-tuned in subsequent joint training. Both ORConvQA and DPR retrievers take the question rewrite as input following the setting in [21]. The only difference between them is that DPR uses a TF-IDF hard negative provided by the dataset in addition to the in-batch negatives. DPR outperforms ORConvQA when the batch size is the same, which indicates that the hard negative play a key role during training. Considering the batch size of ORConvQA in [21] is 64, we also show the retrieval performance, which is improved dramatically as the number of in-batch negatives increases. Our KL-divergence based regularized retriever uses two question forms, forward and backward, for reader effectively improves the performance compared to using BERT. Overall, MICQA achieves the best performance whether with BERT or RoBERTa.

Table 3: Performance of ablation on different components. MICQA refers to the full system.

| Settings | MICQA w/o KL w/o post-ranker w/o curriculum |
|----------|---------------------------------------------|
| F1       | 28.1 27.7 27.8 27.8                     |
| HEQ-Q    | 19.5 19.4 19.4 18.3                     |
| Dev MRR  | 0.4 0.0 0.0 0.0                         |
| Test Rt MRR | 0.6169 0.5324 0.6157 0.6140         |
| Test Rt Recall | 0.3819 0.2968 0.3830 0.3764       |

Table 4: Results of retriever pre-training. B is batch size, Q is the form of question used in training, HD is the number of hard negatives.

| B | Q | HD | Recall@20 | Recall@100 |
|---|---|----|-----------|------------|
| BM25 | / | / | 0.3711 | 0.5100 |
| ORConvQA | 16 | q^w | 0 | 0.1672 | 0.2916 |
|    | 16×4 | q^w/q^w | 0 | 0.3561 | 0.5034 |
| DPR | 16 | q^w | 1 | 0.2395 | 0.3759 |
| ours | 16 | q^w/q^w | 1 | 0.3214 | 0.4689 |
|    | 16 | q^w/q^w | 1 | 0.3690 | 0.5070 |
|    | 32 | q^w/q^w | 1 | 0.4675 | 0.5882 |
5 RELATED WORK

This section briefly summarizes some existing works on open domain question answering and conversational question answering, which are most relevant to this work.

5.1 Open Domain Question Answering

Open-domain question answering (OpenQA) [28] is a task that uses a huge library of documents to answer factual queries. The two-stage design, which included a passage retriever to choose a subset of passages and a machine reader to extract answers, became popular after DrQA [3].

The passage retriever is an important component of OpenQA system since it searches relevant paragraphs for the next stage. Traditional sparse retrieval models, such as TF-IDF or BM25 [25], have been widely adopted as retriever in OpenQA systems [3, 17, 32]. While sparse retrieval cannot handle the case of high semantic correlation with little lexical overlap and it is untrainable, dense passage retrievers have lately gained popularity [8, 11, 15, 24]. In general, the dense retrieval model is a dual-encoder architecture that encodes both the question and the passage individually. Both encoders are trained during the retriever pre-training process. When training with the reader for the QA task, only the question encoder is normally fine-tuned. In order to increase the retrieval impact of dense retrievers, some studies incorporate hard negatives. BM25 top passages which do not contain answers are utilized as hard negatives. [8, 30, 33], which are the top-ranked irrelevant documents given by dense retriever during training.

A contemporary OpenQA system also includes a reader as a key component. Its goal is to deduce the answer to a query from a collection of documents. Existing Readers may be divided into two types: (1) Extractive readers [3, 11, 32], which anticipate an answer span from the retrieved texts, (2) Generative readers [9, 16], which produce natural language replies using sequence-to-sequence (Seq2Seq) models.

5.2 Conversational Question Answering

Conversational Question Answering (CQA) is required to understand the given context and history dialogue to answer the question. As a main type of CQA, Conversational Machine Reading Comprehension (CMRC) [20, 22, 23] does the QA task with text-based corpora. For CMRC, the number of conversational history turns is critical, as context utterances that are relevant to the inquiry are...
valuable, while irrelevant ones may introduce additional noise. For example, [21, 22] makes use of conversation history by including K rounds of history turns. [23] weighs previous conversation rounds based on their contribution to the answer to the current question.

The approaches outlined above rely extensively either on the provided material or a given paragraph to extract or generate answers. However, this is impractical in real world since golden passage is not always available. Open retrieval methods which try to obtain evidence from a big collection, have lately been popular in the CMRC. ORConvQA [21] is the first work proposing three primary modules for open-retrieval CQA: (1) a passage retriever, (2) a passage reranker, and (3) a passage reader. Given a query and its conversational history, the passage retriever retrieves the top K relevant texts from a large-scale corpus. The passage reranker and reader then rerank and read the top texts to discover the correct answer. The research for the conversational OpenQA needs to be further explored. This work tries to improve ORConvQA from multiple aspects including regularizing retriever pre-training, incorporating post-ranking, and curriculum learning.

6 CONCLUSION

This paper proposes multifaceted improvements for the conversational OpenQA task. Concretely, a KL-divergence based regularization is proposed in pre-training for a better question understanding. A post-ranker module is added to realize the joint training of question and passage representations to generate a better passage ranking. A semi-automatic curriculum learning strategy is designed to encourage the reader to find the answer without manually adding golden passages. The experimental evaluation demonstrates the effectiveness of our MICQA.

REFERENCES

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In Proceedings of the 26th Annual International Conference on Machine Learning. 41–48.

[2] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a “siamese” time delay neural network. International Journal of Pattern Recognition and Artificial Intelligence 7, 94 (1993), 669–688.

[3] 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. 1870–1879.

[4] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question Answering in Context. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2174–2184.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2019). 4171–4186.

[6] Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. 2019. Can You Un-pack That? Learning to Rewrite Questions-in-Context. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 5918–5924.

[7] Loyu Gao, Zhiyuan Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2020. Complementing lexical retrieval with semantic residual embedding. arXiv preprint arXiv:2004.13969 (2020).

[8] Kelvin Guo, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. arXiv preprint arXiv:2002.08999 (2020).

[9] Gautier Icabiard and Édouard 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. 874–880.

[10] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPFs. arXiv preprint arXiv:1702.08734 (2017).

[11] Vladimir Karpukhin, Barlas Ogus, 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). 6769–6781.

[12] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL-HLT. 4171–4186.

[13] Bernhard Kratzwald and Stefan Feuerriegel. 2018. Adaptive Document Retrieval for Deep Question Answering. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 576–581.

[14] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. arXiv:1909.11942 [cs.CL].

[15] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. arXiv preprint arXiv:1908.00300 (2019).

[16] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. arXiv preprint arXiv:2005.11401 (2020).

[17] Yankai Liu, Haozhe Ji, Zhiyu Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 1736–1745.

[18] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Yonatan Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019).

[19] Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Knowledge guided text retrieval and reading for open domain question answering. arXiv preprint arXiv:1911.03588 (2019).

[20] Minghui Qu, Xinjing Huang, Cen Chen, Feng Ji, Chen Qu, Wei Wei, Jun Huang, and Yin Zhang. 2021. Reinforced history backtracking for conversational question answering. In Proceedings of the 55th Conference on Artificial Intelligence, AAAI.

[21] Chen Qu, Liu Yang, Cen Chen, Minghui Qu, W Bruce Croft, and Mohit Iyyer. 2020. Open-retrieval conversational question answering. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 539–548.

[22] Chen Qu, Liu Yang, Minghui Qu, W Bruce Croft, Yongfeng Zhang, and Mohit Iyyer. 2019. BERT with history answer embedding for conversational question answering. In Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval. 1133–1136.
[23] Chen Qu, Liu Yang, Minghui Qiu, Yongfeng Zhang, Cen Chen, W Bruce Croft, and Mohit Iyyer. 2019. Attentive history selection for conversational question answering. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 1391–1400.

[24] 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.

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

[26] Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. arXiv preprint arXiv:1602.03609 (2016).

[27] Valentin I Spitkovsky, Hiyan Alshawi, and Dan Jurafsky. 2010. From baby steps to leapfrog: How "less is more" in unsupervised dependency parsing. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. 751–759.

[28] Ellen M Voorhees et al. 1999. The TREC-8 question answering track report. In Trec, Vol. 99. Citeseer, 77–82.

[29] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. 2006. Distance metric learning for large margin nearest neighbor classification. In Advances in neural information processing systems. 1473–1480.

[30] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Pung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold overlook. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In International Conference on Learning Representations (ICLR). https://www.microsoft.com/en-us/research/publication/approximate-nearest-neighbor-negative-contrastive-learning-for-dense-text-retrieval/

[31] Wenhan Xiong, Xiang Li, Srinivas Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, et al. 2020. Answering Complex Open-Domain Questions with Multi-Hop Dense Retrieval. In International Conference on Learning Representations.

[32] Wei Yang, Yiqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-End Open-Domain Question Answering with BERTerini. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics. 72–77.

[33] Jingtao Zhan, Jiaxin Mao, Yagun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing Dense Retrieval Model Training with Hard Negatives. arXiv preprint arXiv:2104.06051 (2021).

[34] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and reading: A comprehensive survey on open-domain question answering. arXiv preprint arXiv:2103.00774 (2021).