Open-Retrieval Conversational Question Answering

Chen Qu[1] Liu Yang[1] Cen Chen[2] Minghui Qiu[3] W. Bruce Croft[1] Mohit Iyyer[1]
[1] University of Massachusetts Amherst [2] Ant Financial [3] Alibaba Group
{chenqu,lyang,croft,miyyer}@cs.umass.edu,chencen.cc@antfin.com,minghui.qmh@alibaba-inc.com

ABSTRACT

Conversational search is one of the ultimate goals of information retrieval. Recent research approaches conversational search by simplified settings of response ranking and conversational question answering, where an answer is either selected from a *given* candidate set or extracted from a *given* passage. These simplifications neglect the fundamental role of retrieval in conversational search. To address this limitation, we introduce an open-retrieval conversational question answering (ORConvQA) setting, where we *learn to retrieve evidence from a large collection* before extracting answers, as a further step towards building functional conversational search systems. We create a dataset, OR-QuAC, to facilitate research on ORConvQA. We build an end-to-end system for ORConvQA, featuring a retriever, a reranker, and a reader that are all based on Transformers. Our extensive experiments on OR-QuAC demonstrate that a learnable retriever is crucial for ORConvQA. We further show that our system can make a substantial improvement when we enable history modeling in all system components. Moreover, we show that the reranker component contributes to the model performance by providing a regularization effect. Finally, further in-depth analyses are performed to provide new insights into ORConvQA.

KEYWORDS

Conversational Question Answering; Open-Retrieval; Conversational Search

ACM Reference Format:
Chen Qu, Liu Yang, Cen Chen, Minghui Qiu, 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 (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401110

1 INTRODUCTION

Conversational search is an embodiment of an iterative and interactive information retrieval (IR) system that has been studied for decades [2, 10, 29]. Due to the recent rise of intelligent assistant systems, such as Siri, Alexa, AliMe, Cortana, and Google Assistant, a growing part of the population is moving their information-seeking activities to voice or text based conversational interfaces. This trend is closely related to the revival of research interest in question answering (QA) and conversational QA (ConvQA). ConvQA can be considered as a simplified setting of conversational search [33]. A significant limitation of this setting is that an answer is either extracted from a *given* passage [34] or selected from a *given* candidate set [53]. This simplification neglects the fundamental role of retrieval in conversational search. To address this issue, we introduce an open-retrieval ConvQA (ORConvQA) setting, where we *learn to retrieve evidence from a large collection* before extracting answers.

Table 1: Comparison of selected QA tasks on the dimensions of open-retrieval (OR), conversational (Conv), information-seeking (IS), and whether motivated by genuine information needs (GIN). The symbol "-" suggests a mixed situation.

Task & Example Data	OR	Conv	IS	GIN
Open-Retrieval QA [11, 26]	✓	✗	-	-
Response Ranking w/ UDC [27, 49, 53]	✗	✓	✓	✓
Conversational MC w/ CoQA [5, 61]	✗	✓	✗	✗
Conversational MC w/ QuAC [18, 34]	✗	✓	✓	✓
ORConvQA w/ OR-QuAC (this work)	✓	✓	✓	✓

We illustrate the importance of ORConvQA by characterizing the task and discussing the considerations of an ORConvQA dataset as follows. A comparison between ORConvQA and related tasks is presented in Table 1.

1. **Open-retrieval**. This is a defining property of ORConvQA. In recent ConvQA datasets [6, 37], the ConvQA task is formulated as a conversational machine comprehension (MC) problem with the goal being to extract or generate an answer from a given gold passage. This setting can be impractical in real-world applications since the gold passage is not always available, or there could be no ground truth answer in the given passage. Instead of being given the passage, a ConvQA system should be able to retrieve candidate passages from a collection. In particular, it is desirable if this retriever is learnable and can be fine-tuned on the downstream ConvQA task, instead of adopting fixed heuristic retrieval functions like TF-IDF or BM25. Moreover, the retrieval process should be open in terms of retrieving from a large collection instead of reranking a small number of passages in a closed set.

2. **Conversational**. Being conversational reflects the interactive nature of a search activity. The important problem of user interaction modeling in IR can be formulated as conversation history modeling in this scenario.

3. **Information-seeking**. An information-seeking conversation typically requires multiple turns of information exchange to allow the seeker to clarify an information need, provide feedback, and ask follow up questions. In this process, answers are revealed to the seeker through a sequence of interactions between the seeker

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401110

and the provider. These answers are generally longer than the entity-based answers in factoid QA.

4. **Genuine information needs**. An information-seeking conversation is closer to real-world scenarios if the seeker is genuinely seeking an answer. In SQuAD [35] and CoQA [37], the seekers' information needs are not genuine because they have access to the passage and thus have the target answer in mind when asking the question. These questions are referred to as "back-written questions" [1] and have been reported to have more lexical overlap with their answers in SQuAD [1]. This undesirable property makes the models learned from such datasets less practical.

To the best of our knowledge, there has not been a publicly available dataset that satisfies all the properties we discussed as shown in Table 1. We address this issue by aggregating existing data to create the *OR-QuAC* dataset. The QuAC [6] dataset offers information-seeking conversations that are collected with no seekers' prior knowledge of the passages. We extend QuAC to an open-retrieval setting by creating a collection of over 11 million passages using the whole Wikipedia corpus. Another important resource used in our aggregation process is the CANARD dataset [15] that offers context-independent rewrites of QuAC questions. Some initial questions in QuAC conversations are underspecified. This makes conversation difficult to interpret in an open-retrieval setting. We make these dialogs self-contained by replacing the *initial* question in a conversation with its rewrite from CANARD. Our data has 5,644 dialogs with 40,527 questions. We release OR-QuAC to the community to facilitate research on ORConvQA.

In addition to proposing ORConvQA and creating the OR-QuAC dataset, we develop a system for ORConvQA following previous work on open-retrieval QA [26]. Our end-to-end system features a retriever, a reranker, and a reader that are all based on Transformers [45]. We enable history modeling in all components by concatenating history questions to the current question. The passage retriever first retrieves the top K relevant passages from the collection given a question and its history. The reranker and reader then rerank and read the top passages to produce an answer. The training process contains two phases, a pretraining phase for the retriever and a concurrent learning phase for all system components.

Specifically, our retriever adopts a dual-encoder architecture [1, 11, 21, 26] that uses separate ALBERT [24] encoders for questions and passages. The question encoder also encodes conversation history. After being pretrained, the passage encoder is frozen and encodes all passages in the collection offline. The reranker and the reader share the same BERT [12] encoder. It encodes the input sequence of a concatenation of the question, history, and each relevant passage to contextualized representations for reranking and answer extraction. We incorporate shared-normalization [8] in our system to enable comparison among the candidate passages. In the concurrent learning phase, we encode the question and the history to dense vectors with the question encoder for an efficient retrieval with maximum inner product search (MIPS) [19, 38]. The top retrieved passages are fed to the reranker and reader for a concurrent learning of all model components.

We conduct extensive experiments on our OR-QuAC dataset. First, we show that our system without any history information has comparable performance with a conversational version of BERTserini [56] that considers history. This improvement demonstrates the importance of a learnable retriever in ORConvQA. We further show that our system can make a substantial improvement when we enable history modeling in all system components. Moreover, we conduct in-depth analyses on model ablation and configuration to provide insights for the ORConvQA task. We show that our reranker component contributes to the model performance by providing a regularization effect. We also demonstrate that the initial question of each dialog is crucial for our system to understand the user's information need. Our code and data are available for research purposes.[1]

2 RELATED WORK

Our work is closely related to several research topics, including QA, open domain QA, ConvQA, and conversational search. We mainly discuss retrieval based methods since they tend to offer more informative responses [53] and thus better fit for information-seeking tasks than generation based methods.

Question Answering. One of the first modern reformulations of the QA task dates back to the TREC-8 Question Answering Track [46]. Its goal is to answer 200 fact-based, short-answer questions by leveraging a large collection of documents. A retrieval module is crucial in this task to retrieve relevant passages for answer extraction. As an increasing number of researchers in the natural language processing (NLP) community moving their focus to answer extraction and generation methods, the role of retrieval has been gradually overlooked. As a result, many popular QA tasks and datasets either follow an answer selection setting [16, 47, 57] or a machine comprehension setting [23, 35, 36, 44]. In real-world scenarios, it is less practical to assume we are given a small set of candidate answers or a gold passage. Therefore, in this work, we make the retrieval component as one of our focuses in the task formulation and model architecture.

Open Domain Question Answering. In contrast to the tasks that offer a pre-selected passage for answer extraction, open domain QA tasks provide the model with access to a large corpus [13] or at least a set of candidate documents for each question [9, 13, 14, 20, 28]. DrQA [4] and BERTserini [56] present an end-to-end open domain QA system by using a TF-IDF/BM25 retriever and a neural reader. Some previous work [17, 22, 25, 48] learns to rerank or select from a closed set of passages for open domain QA. These methods may not scale well to an open-retrieval setting. Recently, Lee et al. [26], Das et al. [11], and Karpukhin et al. [21] adopt a dual-encoder architecture to construct a learnable retriever and demonstrate their methods are scalable to large collections. ReQA [1] also uses a similar architecture to retrieve sentence-level answers directly. Although these works are limited to single turn QAs, they are valuable resources for us to study how to extend ConvQA to an open-retrieval setting.

Conversational Question Answering. Similar to the answer selection and MC tasks in single-turn QAs, existing ConvQA research is mostly limited to response ranking [27, 39, 49–51, 53–55] and conversational MC [5, 6, 18, 33, 34, 37, 59, 61], where the role of retrieval is also neglected. Open-retrieval is particularly important to ConvQA since the answers of the questions from the same dialog may not necessarily come from the same passage. The model needs

[1] https://github.com/prdwb/orconvqa-release

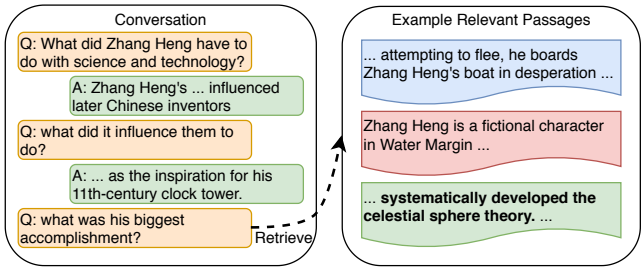


Figure 1: A partial OR-QuAC dialog and example relevant passages retrieved from the collection by TF-IDF.

to learn to retrieve passages for each dialog question. Another challenge is to investigate how to enable history modeling not only in the reader but also in the retriever. Moreover, there are no existing datasets that are suitable to study ORConvQA. Therefore, we tackle these research questions in this work.

Conversational Search. While the concept of conversational search can be traced back to research [2, 10, 29] from decades ago, recent years have witnessed its revival. In addition to the ConvQA work mentioned above, researchers are also actively working on other conversation tasks, including conversational recommendation and product search [3, 60], user intent prediction [32], and question retrieval [52]. Another rich body of work targets the user-oriented aspect [7, 30, 31, 40–43] for conversational information seeking. Our work extends ConvQA to an open-retrieval setting as another fundamental step towards conversational search.

3 THE OR-QUAC DATASET

The OR-QuAC dataset enhances QuAC by adapting it to an open-retrieval setting. It is an aggregation of three existing datasets: (1) the QuAC dataset [6] that offers information-seeking conversations, (2) the CANARD dataset [15] that consists of context-independent rewrites of QuAC questions, and (3) the Wikipedia corpus that serves as the knowledge source of answering questions. An example of OR-QuAC is presented in Figure 1. We will describe the data construction process in the following sections.

3.1 Self-Contained Information-seeking Dialogs

The QuAC (Question Answering in Context) dataset [6] is designed for modeling information-seeking conversations. It consists of real human-human dialogs between an information seeker and an information provider. The seeker tries to learn about a hidden Wikipedia text by asking a sequence of freeform questions. She/he only has access to the title and a summary of the article. This simulates a genuine information need. The provider answers each question by indicating a short span of the given passage. This dataset poses unique challenges because its questions are more open-ended, sometimes unanswerable, or only meaningful within the dialog context [6].

A drawback of QuAC is that many dialogs are not self-contained. This is typically caused by incomplete initial questions. A QuAC dialog is motivated by a general and underlying information need. During the data collection process, this information need is provided to both the seeker and provider before initiating the dialog. Therefore, the seeker might not necessarily reiterate this information need when asking the first question. For example, a seeker in QuAC is instructed to learn about *Zhang Heng*, a Chinese polymathic scientist. The very first question the seeker asked was "*what did he have to do with science and technology?*". Such underspecified and ambiguous initial questions become an issue in the open-retrieval setting because they make the conversation difficult to interpret.

We tackle this issue by replacing *initial* questions in QuAC with their context-independent *rewrites* provided by the CANARD dataset. For example, the rewrite for the previously mentioned question is "*What did Zhang Heng have to do with science and technology?*". We do the replacement for the first questions only. This makes a dialog self-contained while keeping the history dependencies within the dialog untouched.

CANARD covers about half of the released QuAC questions. Since the QuAC test set is not publicly available, they use QuAC's development set as their test set and 10% of QuAC's training set as their development set [15]. We follow the data split of CANARD. QuAC questions that not in CANARD are discarded. The data statistics of our derived dataset, OR-QuAC, are presented in Table 2.

Table 2: Data statistics of the OR-QuAC dataset.

Items	Train	Dev	Test
# Dialogs	4,383	490	771
# Questions	31,526	3,430	5,571
# Avg. Question Tokens	6.7	6.6	6.7
# Avg. Answer Tokens	12.5	12.6	12.2
# Avg. Questions / Dialog	7.2	7.0	7.2
# Min/Avg/Med/Max History Turns / Question	0/3.4/3/11	0/3.3/3/11	0/3.4/3/11

3.2 Collection

We use the whole Wikipedia corpus to construct a collection since passages in QuAC are from Wikipedia. We use the English Wikipedia dump from 10/20/2019.[2] The Wikipedia passages in QuAC were downloaded via PetScan[3] [6], and thus, the exact date for the data dump is unavailable. Therefore, we use the latest data dump instead of trying to match the date of QuAC. We then use the WikiExtractor[4] to extract and clean text from the data dump, resulting in over 5.9 million Wikipedia articles. After this, we split the articles into chunks with at most 384 wordpieces using the tokenizer of BERT, following Lee et al. [26]. The split is done greedily while preserving sentence boundaries. These chunks are referred to as *passages*. Less than 0.5% of known answers are split into different passages. Their corresponding questions are considered as unanswerable during training. We do the split to make the passages fit for Transformer based retrievers and readers. Moreover, Yang et al. [56] reported that the paragraph level is the best granularity for an end-to-end retrieve-and-read framework compared to the article and sentence levels. They believe the reason is that an article may contain non-relevant content that distracts the reader while a sentence may lack context information. For an open-retrieval setting, we prefer passage-level retrieval over article-level since a full article would be harder to represent with a fixed-length dense vector.

[2] https://dumps.wikimedia.org/enwiki/20191020/ [3] http://petscan.wmflabs.org/
[4] https://github.com/attardi/wikiextractor

Since the paragraphs in QuAC may not be exactly the same as those in the Wikipedia dump given the difference in the dates of the dumps, we conduct the same split process for QuAC paragraphs and replace the Wikipedia passages with QuAC passages that have the same article titles. The positions of the ground truth answer spans are mapped to the new passages. The resulting collection has over 11 million passages for retrieval.

Due to the synthetic nature of this dataset, the answers of the questions in the same dialog are distributed in the same section of text. In real world, questions and answers in a dialog may be distributed at different locations of the corpus. This is a limitation of our dataset.

4 AN END-TO-END ORCONVQA SYSTEM

In this section, we first formally define the task of open-retrieval conversational QA. We then describe our end-to-end system that deals with this task and explain the intuitions behind it.

4.1 Task Definition

The ORConvQA task is defined as follows. Given the k-th question q_k in a conversation, and all history questions $\{q_i\}_{i=1}^{k-1}$ preceding q_k, the task is to predict an answer a_k for q_k using a passage collection C. In an extractive setting, a_k is a text span of a passage in C. We do not assume we have access to ground truth history answers since it is impractical in real-world scenarios.

Extractive models are trained on the supervision signals of the position of a span in the gold passage. Previous works [4, 11, 26] present a distantly-supervised setting, where they only have access to QA pairs without gold passages. This setting heavily relies on a heuristic that a positive passage should contain an exact match of the known answer. Short and entity-based answers can often be discovered in multiple passages, meaning that positive passages are highly substitutable. In information-seeking conversations that are motivated by genuine information needs, however, the answers are typically much longer. For example, QuAC answers have 12 tokens on average while SQuAD and CoQA answers have 3 [37]. It is common that the retrieved passages do not contain exact matches of the known answers, making many training examples useless. To tackle this, we adopt a fully-supervised setting: we assume we have access to gold passages so that we can include them if they are not present in the retrieval results and use the ground-truth answer spans. This is done at *training* time only. Although this is a limitation, it does not conflict with the learnable retriever we promote. We will work on a weak supervision method that is suitable for information-seeking conversations in our future work.

4.2 Model Overview

We now present an end-to-end system that deals with the ORConvQA task described in Section 4.1. Our system consists of three major components, a passage retriever, a passage reranker, and a passage reader. The reranker and reader are based on the same encoder. All components are learnable. As described in Figure 2, the passage retriever first retrieves top-K relevant passages from the collection given a question and its history. The passage reranker and reader then rerank and read the top passages to produce an answer. History modeling is enabled in all components. We will describe each component in detail in the following sections.

4.3 Passage Retriever

We present the retriever module in the upper-left part of Figure 2. We follow previous research [1, 11, 26] by using a dual-encoder architecture to construct a learnable retriever. This architecture features separated encoders for questions and passages. The retriever score is then defined as the dot product of the hidden representations of a question and a passage. We use two ALBERT [24] models for both encoders. ALBERT is a lite BERT [12] model for learning bidirectional language representations from Transformers [45]. It reduces the parameters of BERT by cross-layer parameter sharing and embedding parameters factorization [24].

Given all available history questions $\{q_i\}_{i=1}^{k-1}$, we first identify those that are in a history window with the size w. These questions are denoted as $\{q_i\}_{i=k-w}^{k-1}$. We then construct a concatenation of $\{q_i\}_{i=k-w}^{k-1}$ and q_k. We prepend the initial question q_1 of the conversation to the concatenation if q_1 is not already included. The initial question q_1 typically contains an information need that is pertinent to the entire conversation as explained in Section 3.1. The reformatted question for the retriever is denoted as q_k^{rt}. For an ALBERT based question encoder, the input sequence would be "`[CLS]` q_1 `[SEP]` q_{k-w} `[SEP]` $\cdots$ `[SEP]` q_{k-1}`[SEP]` q_k `[SEP]`". All questions are in the same segment. `[CLS]` and `[SEP]` are special tokens introduced in BERT [12]. We then take the `[CLS]` representation and project it to a 128-dimensional vector as the question representation following Lee et al. [26]. Formally,

$$v_q = W_q \, \text{ALBERT}_q(q_k^{rt})\texttt{[CLS]} \tag{1}$$

where ALBERT_q is the question encoder, W_q is the projection matrix for the question `[CLS]` representation, and $v_q \in \mathbb{R}^{1\times 128}$ is the final question representation enhanced with history information. We then follow the same scheme to obtain the passage representation for a passage p_j:

$$v_p = W_p \, \text{ALBERT}_p(p_j)\texttt{[CLS]} \tag{2}$$

where p_j is a passage in the collection, ALBERT_p is the passage encoder, W_p is the projection matrix for the passage `[CLS]` representation, and $v_p \in \mathbb{R}^{1\times 128}$ is the final passage representation. Finally, the retrieval score is computed as

$$S_{rt}(q_k^{rt}, p_j) = v_q v_p^\top \tag{3}$$

4.4 Passage Reader/Reranker

Given the current question q_k, history questions $\{q_i\}_{i=1}^{k-1}$, the history window size w, and one of the retrieved passages p_j, the passage reader predicts an answer span within the passage. In contrast to Lee et al. [26] and Yang et al. [56], we introduce reranking into this process with little additional cost. Our reader mostly follows the standard architecture of a BERT based MC model [12]. We enhance this model by applying the shared-normalization mechanism proposed by Clark and Gardner [8] to enable comparison across all retrieved passages for a question. Similar mechanisms are also adopted by Yang et al. [56] and Lee et al. [26].

4.4.1 ***Encoder.*** The reader and reranker share the same BERT encoder. Similar to the retriever, we first construct a reformatted

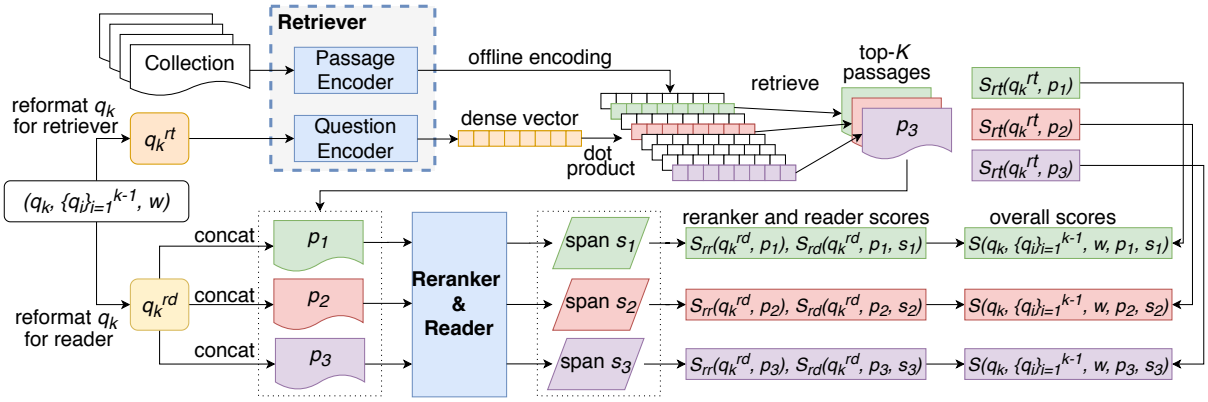


Figure 2: Architecture of our end-to-end ORConvQA model. The input is the current question q_k, all history questions $\{q_i\}_{i=1}^{k-1}$, and a history window size w. The retriever first retrieves top-K relevant passages from the collection and generates retriever scores S_{rt}. The reranker and reader then rerank and read the top passages to produce an answer span for each passage and generate reranker and reader scores, S_{rr} and S_{rd}. The system outputs the answer span with the highest overall score S.

question by concatenating history questions within a history window and the current question. We do not additionally prepend the initial question because the conversation is considered to be grounded to p_j. The reformatted question for the reader is denoted as q_k^{rd}. We then concatenate a retrieved passage to form the input sequence for the BERT model. Specifically, the input sequence (q_k^{rd}, p_j) is "[CLS] q_{k-w} [SEP] $\cdots$ [SEP] q_{k-1}[SEP] q_k [SEP] p_j [SEP]", with q_k^{rd} and p_j in different segments. The BERT model then generates contextualized representations for all tokens in the input sequence:

$$v_{[m]} = \text{BERT}((q_k^{rd}, p_j))[m] \tag{4}$$

where $v_{[m]}$ is the representation for the m-th token in the input sequence. We also need the sequence representation obtained by

$$v_{[CLS]} = W_{[CLS]}\,\text{BERT}((q_k^{rd}, p_j))[CLS] \tag{5}$$

where $W_{[CLS]}$ is a projection for the [CLS] representation to obtain the sequence representation $v_{[CLS]}$ following Devlin et al. [12].

4.4.2 ***Reranker.*** As shown in Figure 2. The reranker components conduct a listwise reranking of the top retrieved passages. The reranking task provides more supervision signals to fine-tune the BERT encoder. The representation learning of the encoder also benefits from a regularization effect for optimizing for multiple tasks. Moreover, the reranking task adds little additional cost to the training process because representations for all tokens, including the [CLS] token, are generated with vectorization in a Transformer architecture. Specifically, we learn a reranking vector $\mathbf{W}_{rr}$ to project the sequence representation $v_{[CLS]}$ to a reranking score S_{rr}:

$$S_{rr}(q_k^{rd}, p_j) = \mathbf{W}_{rr} v_{[CLS]} \tag{6}$$

4.4.3 ***Reader.*** The reader predicts an answer span by computing scores of each token being the start token and the end token. We learn two sets of parameters, a start vector $\mathbf{W}_s$ and an end vector $\mathbf{W}_e$, to project token representations to start and end scores:

$$S_s(q_k^{rd}, p_j, [m]) = \mathbf{W}_s v_{[m]} \quad , \quad S_e(q_k^{rd}, p_j, [m]) = \mathbf{W}_e v_{[m]} \tag{7}$$

where $S_s(q_k^{rd}, p_j, [m])$ and $S_e(q_k^{rd}, p_j, [m])$ are the scores for the m-th token being the start and end tokens of the answer span. The reader score and overall score will be computed at inference time in Section 4.6.

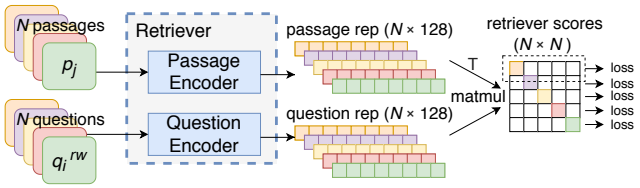


Figure 3: Retriever pretraining.

4.5 Training

Our training procedure contains two phases. The first is the retriever pretraining phase, followed by the concurrent learning phase of the retriever (question encoder), reranker, and reader.

4.5.1 ***Retriever Pretraining.*** We follow previous work [26] to pretrain the retriever so that it gives a reasonable performance in the concurrent learning phase.

In Section 4.3, we mentioned that history modeling is enabled in the retriever by prepending history questions. The history window size w is a hyper-parameter and is tunable. In the pretraining phase, however, we would like to train a uniform retriever for every single history window size. Therefore, we use the rewrite in CANARD q_i^{rw} as the reformatted question for a question q_i in the pretraining phase. We will mitigate the question mismatch issue by fine-tuning the question encoder in the concurrent learning phase.

The pretraining process of the retriever is described in Figure 3. Given a batch of N question representations $V_q \in \mathbb{R}^{N\times 128}$ and their gold passage representations $V_p \in \mathbb{R}^{N\times 128}$, we obtain the retrieval scores for the batch by

$$\mathbf{S}_{rt}(V_q, V_p) = V_q V_p^\top \tag{8}$$

where $\mathbf{S}_{rt}(V_q, V_p) \in \mathbb{R}^{N\times N}$. The element $S_{i,j}$ in the i-th row and j-th column of $\mathbf{S}_{rt}(V_q, V_p)$ represents $S_{rt}(q_i^{rw}, p_j)$. The objective is

to maximize the probability of the gold passage for each question:

$$P_{rt}(p_j|q_i^{rw}) = \frac{\exp(S_{rt}(q_i^{rw}, p_j))}{\sum_{j'=1}^{N} \exp(S_{rt}(q_i^{rw}, p_{j'}))} \tag{9}$$

In other words, the passage set $\{p_j\}_{j=1}^{N} - \{p_i\}$ is considered as randomly sampled negative passages for q_i. The pretraining loss for this batch is then defined as follows.

$$\mathcal{L}_{\text{pretrain}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} \mathbb{1}\{j=i\} \log P_{rt}(p_j|q_i^{rw}) \tag{10}$$

Lee et al. [26] suggest that it is crucial to set the batch size N to a large number because it makes the pretraining task more difficult and closer to what the retriever observes at test time. Therefore, we use two ALBERT models as the question encoder and the passage encoder. This doubles the batch size compared to that of using BERT models. The ALBERT models are fine-tuned.

We then encode all passages in the collection C offline with the passage encoder and obtain a set of passage vectors. Finally, we use Faiss[5], a library for efficient similarity search of dense vectors, to create an index for maximum inner product search. Retrieval is performed on a GPU during concurrent learning for faster training.

4.5.2 ***Concurrent Learning of the Retriever, Reranker, and Reader.*** As indicated in Figure 2, given the current question q_k, the history questions $\{q_i\}_{i=1}^{k-1}$, and the history window size w, we obtain the reformatted question for the retriever q_k^{rt} and the reader q_k^{rd}. We first obtain the question representation of q_k^{rt} using the question encoder in Equation 1. We then retrieve the top K_{rd} passages for the reader from the passage collection using the index we created offline. This set of top passages is denoted as TK_{rd}. The number of negative samples for retriever is limited by the CUDA memory in the retriever pretraining phase. In the concurrent learning phase, we can use a relatively large amount of negative samples to fine-tune the retriever at a low cost since all passages have been encoded offline. Therefore, we also retrieve the top K_{rt} passages, where $K_{rt} > K_{rd}$, for an aggressive update of the retriever following Lee et al. [26]. This set of passages is denoted as TK_{rt}. If the gold passage of q_k is not present in TK_{rt} or TK_{rd}, we manually include it in the retrieval results. Formally, the retriever loss to fine-tune the question encoder in the retriever is defined as follows.

$$\mathcal{L}_{rt} = -\sum_{p_j \in TK_{rt}} \mathbb{1}\{j=j_{rt}\} \log P_{rt}(p_j|q_k^{rt}) \tag{11}$$

where j_{rt} is the position of the gold passage in TK_{rt}.

Passages in TK_{rd} are then fed into the reader/reranker module. This module conducts reading and reranking simultaneously. For every passage $p_j \in TK_{rd}$, we obtain a reranking score $S_{rr}(q_k^{rd}, p_j)$ following Equation 6. We then compute the reranking probability and the reranking loss as follows.

$$P_{rr}(p_j|q_k^{rd}) = \frac{\exp(S_{rr}(q_k^{rd}, p_j))}{\sum_{p_{j'} \in TK_{rd}} \exp(S_{rr}(q_k^{rd}, p_{j'}))} \tag{12}$$

$$\mathcal{L}_{rr} = -\sum_{p_j \in TK_{rd}} \mathbb{1}\{j=j_{rd}\} \log P_{rr}(p_j|q_k^{rd}) \tag{13}$$

where j_{rd} is the position of the gold passage in TK_{rd}.

For the reader component, a standard BERT based machine comprehension model uses the cross entropy loss to maximize the probability of the true start and end tokens among all tokens in the given passage. Different from that, we apply the shared-normalization mechanism [8] to this step to maximize the probabilities of the true start and end tokens among all tokens from TK_{rd}. This makes the model produce start and end scores that are comparable across passages. The passages are encoded independently, and the shared-normalization is applied to all passages at the last step. For a passage $p_j \in TK_{rd}$, we obtain a start score $S_s(q_k^{rd}, p_j, \texttt{[m]})$ for every token `[m]` in the input sequence. The training loss for the start token is then defined as follows.

$$P_s(q_k^{rd}, p_j, \texttt{[m]}) = \frac{\exp(S_s(q_k^{rd}, p_j, \texttt{[m]}))}{\sum_{p_{j'} \in TK_{rd}} \sum_{\texttt{[m']} \in (q_k^{rd}, p_{j'})} \exp(S_s(q_k^{rd}, p_{j'}, \texttt{[m']}))} \tag{14}$$

$$\mathcal{L}_s = -\sum_{p_j \in TK_{rd}} \sum_{\texttt{[m]} \in (q_k^{rd}, p_j)} \mathbb{1}\{j=j_{rd}, \texttt{[m]}=\texttt{[S]}\} \log P_s(q_k^{rd}, p_j, \texttt{[m]}) \tag{15}$$

where `[S]` is the true start token in the gold passage. For unanswerable questions, we set the start and end tokens to `[CLS]`. The BERT encoder is fine-tuned. The loss function of the end token $\mathcal{L}_e$ is defined in the same way. The reader loss is computed as follows.

$$\mathcal{L}_{rd} = \frac{1}{2}(\mathcal{L}_s + \mathcal{L}_e) \tag{16}$$

Finally, the concurrent learning loss is computed as:

$$\mathcal{L} = \mathcal{L}_{rt} + \mathcal{L}_{rr} + \mathcal{L}_{rd} \tag{17}$$

Although the gradients of the reader/reranker do not back propagate to the retriever, we train these modules concurrently so that the reader/reranker can benefit from seeing more negative passages due to a dynamically changing set of retrieved passages TK_{rd}.

4.6 Inference

Given the current question q_k, the history questions $\{q_i\}_{i=1}^{k-1}$, and the history window size w, we follow the same process in the concurrent learning phase to retrieve a set of relevant passages TK_{rd}. Note we do not manually include the gold passage in TK_{rd} at inference time. For a passage $p_j \in TK_{rd}$, we obtain the retriever score $S_{rt}(q_k^{rt}, p_j)$ and the reranker score $S_{rr}(q_k^{rd}, p_j)$ following Equations 3 and 6. We then follow Devlin et al. [12] to obtain the reader score using the start score $S_s(q_k^{rd}, p_j, \texttt{[m]})$ and the end score $S_s(q_k^{rd}, p_j, \texttt{[m]})$ in Equation 7 as follows.

$$S_{rd}(q_k^{rd}, p_j, s) = \max_{[\texttt{m}_\texttt{s}], [\texttt{m}_\texttt{e}] \in (q_k^{rd}, p_j)} S_s(q_k^{rd}, p_j, [\texttt{m}_\texttt{s}]) + S_e(q_k^{rd}, p_j, [\texttt{m}_\texttt{e}]) \tag{18}$$

where s is the answer span with the start token $[\texttt{m}_\texttt{s}]$ and end token $[\texttt{m}_\texttt{e}]$. To ensure tractability, we only consider the top 20 spans following convention [12]. Invalid predictions, including the cases where the start token comes after the end token, or the predicted span overlaps with the question part of the input sequence, are discarded. Finally, the overall score is defined as a function of the current question q_k, its history questions $\{q_i\}_{i=1}^{k-1}$, a history window size w, a retrieved passage p_j, and a answer span s as in Figure 2:

$$\begin{aligned} S(q_k, \{q_i\}_{i=1}^{k-1}, w, p_j, s) = & S_{rt}(q_k^{rt}, p_j) + \\ & S_{rr}(q_k^{rd}, p_j) + \\ & S_{rd}(q_k^{rd}, p_j, s) \end{aligned} \tag{19}$$

The system outputs the answer span that has the largest overall score for each question in a conversation.

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

5 EXPERIMENTAL SETUPS

We now describe our experimental setups, including competing methods, evaluation metrics, and implementation details.

5.1 Competing Methods

To the best of our knowledge, there is no published work tackling the ORConvQA problem that we describe in Section 4.1. There is, however, a rich body of work on single-turn open-domain QA, led by DrQA [4]. We can adapt such methods to a conversational setting by using the same history modeling method in our system. Given the effort to adapt such models to ORConvQA, we only compare to the original DrQA and the best model that we are aware of, BERTserini [56]. To be specific, the competing methods are:

- **DrQA** [4]. This model uses a TF-IDF retriever and an RNN based reader. We train this model on OR-QuAC dialogs with gold passages. At test time, the passages are retrieved with the retriever. This setting is consistent with DrQA's original setting. We do not use its distantly-supervised setting since we would like to adopt full supervision for all competing methods in this work. We start from their open-sourced implementation on GitHub.[6]
- **BERTserini** [56]. This model uses a BM25 retriever from Anserini[7] and a BERT reader. Their BERT reader is similar to ours, except that it does not support reranking and thus cannot benefit from multi-tasking learning. They study the granularity of retrieval, including article, paragraph, and sentence. They conclude that retrieval on a paragraph level gives the best overall performance. We only compare to the paragraph retrieval setting since it is the best and is consistent with our passage retrieval setting. We use the top 5 passages for the reader to be consistent with our setup. This baseline is our implementation since BERTserini's source code was not available at the time of our submission.
- **ORConvQA without history** (Ours w/o hist.). This is our model described in Section 4 with the history window size $w = 0$. Note that the first question of a dialog is still included in the reformatted question for the retriever, as described in Section 4.3. This model is our adaptation of the open-retrieval QA framework [26] to a conversational setting. We use a more direct and resource-efficient retriever pretraining method that is suitable for ConvQA. We also enable reranking in the reader component.
- **ORConvQA** (Ours). This is our full model described in Section 4.

We adapt DrQA and BERTserini to a conversational setting using the same history modeling method in our model. It involves prepending history questions for reformatted questions for the retriever and the reader. For these models and our ORConvQA model, the history window size w is tuned on the development set. We report their performance under the best history setting.

5.2 Evaluation Metrics

The word-level **F1** and the human equivalence score (**HEQ**) are two metrics provided by the QuAC challenge to evaluate ConvQA systems. F1 measures the overlap of the predicted answer span and the ground truth answer span. This is our most important metric since it evaluates the overall performance of the system. HEQ computes the percentage of examples for which system F1 exceeds or matches human F1. It measures whether a system can give answers as good as an average human. This metric is computed on a question level (HEQ-Q) and a dialog level (HEQ-D).

In addition to F1 and HEQ, we also use the Mean Reciprocal Rank (**MRR**) and **Recall** to evaluate the retrieval performance for the retriever and reranker. The reciprocal rank of a query is the inverse of the rank of the first positive passage in the retrieved passages. MRR is the mean of the reciprocal ranks of all queries. This metric is computed for both the retriever and reranker. MRR is a reflection of how well these two components contribute to the overall score in Equation 19. Recall is the fraction of the total amount of relevant passages that are retrieved. There is only one positive passage for each question in the training and development sets. In comparison, there could be more than one positive passage for a testing question since there are multiple reference answers per question provided by QuAC. Recall is computed for the retriever only since reranking does not impact this measure. This metric reflects whether the retriever can provide reasonable retrieval performance for the rest of the system. All retrieval metrics are computed for the top 5 passages that are retrieved for the reader/reranker.

5.3 Implementation Details

Our models are implemented with PyTorch[8] and the open-source implementation of ALBERT and BERT by Hugging Face.[9]

5.3.1 ***Retriever and Pretraining.*** We use two ALBERT Base (V1) models for the question and passage encoders. We set the max sequence length of the question encoder to 128, that of the passage encoder to 384, the training batch size to 16 per GPU, the number of training epochs to 12, and the learning rate to 5e-5. Models are trained with 4 NVIDIA TITAN X GPUs. We create a smaller collection to evaluate the retrieval performance by collecting the top 50 documents retrieved by TF-IDF for development questions. This allows us to do model selection in a scenario that is closer to how the retriever operates during concurrent learning. We save checkpoints every 5,000 steps and evaluate on the development questions to select the best model for concurrent learning. The pretraining time for the retriever is 2.5 hours.

5.3.2 ***Reranker, Reader, and Concurrent Learning.*** We use the BERT Base (Uncased) model. We set the max sequence length to 512, the max question length to 125 (so that the passage length is at least 384 after accounting for a `[CLS]` and two `[SEP]` tokens), the training batch size to 2, the number of training epochs to 3, and the learning rate to 5e-5. We retrieve top 5 passages for the reader. We tune the number of passages to update retriever K_{rt} and the history window size w in Section 6.3. Models are trained with a NVIDIA TITAN X GPU. We take advantage of another TITAN X card for faster MIPS. All passage representations in our collection occupy 7.2 GB of CUDA memory. We save checkpoints every 5,000 steps and evaluate on the development set to select the best model for the test set. The time for concurrent learning is 20.0 hours.

For all model components, we use half precision for training as suggested in the Hugging Face repository to alleviate CUDA memory consumption. The warm up portion of the learning rate is 10% of the total steps.

[6] https://github.com/facebookresearch/DrQA [7] http://anserini.io/

[8] https://pytorch.org/ [9] https://github.com/huggingface/transformers

Table 3: Main evaluation results. "Rt" and "Rr" refers to "Retriever" and "Reranker". ‡ means statistically significant improvement over the strongest baseline with $p < 0.05$.

Settings		DrQA	BERTserini	Ours w/o hist.	Ours
Dev	F1	4.5	19.3	24.0	**26.9**‡
	HEQ-Q	0.0	14.1	15.2	**17.5**
	HEQ-D	0.0	**0.2**	**0.2**	**0.2**
	Rt MRR	0.1151	0.1767	0.4012	**0.4286**‡
	Rr MRR	N/A	N/A	0.4472	**0.5209**‡
	Rt Recall	0.2000	0.2656	0.5271	**0.5714**‡
Test	F1	6.3	26.0	26.3	**29.4**‡
	HEQ-Q	0.1	20.4	20.7	**24.1**
	HEQ-D	0.0	0.1	0.4	**0.6**
	Rt MRR	0.1574	0.1784	0.1979	**0.2246**‡
	Rr MRR	N/A	N/A	0.2702	**0.3127**‡
	Rt Recall	0.2253	0.2507	0.2859	**0.3141**‡

Table 4: Results of ablation studies. "Rt" and "Rr" refers to "Retriever" and "Reranker" respectively. ‡ and † means statistically significant performance *decrease* compared to the full system with $p < 0.05$ and $p < 0.1$ respectively.

Settings		Full system	w/o rerank	w/o learned retriever	w/o first q for retriever
Dev	F1	**26.9**	25.9†	17.1‡	24.6‡
	HEQ-Q	**17.5**	16.8	11.1	15.5
	HEQ-D	0.2	0.2	0.0	**0.4**
	Rt MRR	**0.4286**	0.4031‡	0.1162‡	0.3937‡
	Rr MRR	**0.5209**	N/A	0.1895‡	0.4674‡
	Rt Recall	**0.5714**	0.5411‡	0.2032‡	0.5122‡
Test	F1	**29.4**	27.7‡	24.7‡	27.1‡
	HEQ-Q	**24.1**	22.2	18.1	21.3
	HEQ-D	0.6	0.9	0.5	**1.0**
	Rt MRR	**0.2246**	0.2166‡	0.1603‡	0.2092‡
	Rr MRR	**0.3127**	N/A	0.2130‡	0.2870‡
	Rt Recall	**0.3141**	0.3059†	0.2270‡	0.2918‡

6 EVALUATION RESULTS

In this section, we present our evaluation results, ablation studies on system components, and more analyses on history window size and the number of passages to fine-tune the retriever.

6.1 Main Evaluation Results

We report the main evaluation results in Table 3. We tune the history window size w for all models that consider history and report their performances under the best history setting. The best history settings for DrQA, BERTserini, and Ours are w =5, 2, and 6 respectively. We summarize our observations as follows:

(1) We observe that DrQA has poor performance. The main reason for this lies in the reader component. The RNN based reader in DrQA cannot produce representations that are as good as the readers based on a pretrained BERT in the rest of the competing models. More importantly, the DrQA reader cannot handle unanswerable questions natively.
(2) BERTserini has a significant improvement over DrQA and serves as a much stronger baseline. It addresses the issues in DrQA by using a BERT reader that can handle unanswerable questions. BM25 in Anserini also gives better retrieval performance.
(3) Our model without any history manages to perform on par with BERTserini that considers history on the test set. In particular, our learned retriever achieves higher performance on retrieval metrics. Since our reader is similar to that of BERTserini, the overall performance gain mostly comes from our learned retriever. This verifies the observation in Lee et al. [26] in a conversational setting that a learned retriever is crucial if the information-seeker is genuinely seeking an answer. The margins are substantially larger on the development set, presumably because the best pretrained retriever model is selected based on the development performance.
(4) Our model with history obtains statistically significant improvement over the strongest baseline with $p < 0.05$ tested by the Student's paired t-test. This demonstrates the effectiveness of our model. This also indicates that incorporating conversation history is essential for ORConvQA, as expected. More analyses on the history window size are presented in Section6.3.1. In addition, we observe that the reranker consistently outperforms the retriever. This suggests that although reranking is more expensive as it jointly models the question and the passage, it enjoys better performance than the retriever that models the question and the passage separately.

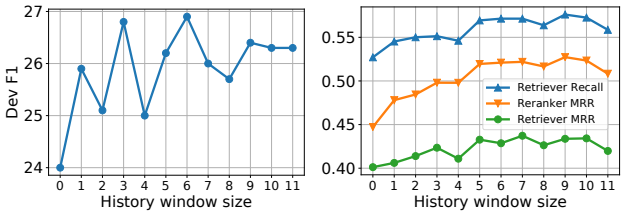
6.2 Ablation Studies

Section 6.1 has shown the effectiveness of our model. This model performance is closely related to several design choices we made. In this section, we conduct ablation studies on our best model in Table 3 to investigate the contributions of each design choice. Specifically, we have three ablation settings as follows.

- **ORConvQA w/o reranker**. We introduce reranking to the system as one of the differences from previous works [11, 26]. In this ablation setting, we remove the reranking loss in Equation 17 so that the encoder in the reader is not fine-tuned by the reranking objective. Naturally, we also do not use the reranking score in the overall score in Equation 19.
- **ORConvQA w/o learned retriever**. We replace our learned retriever with DrQA's TF-IDF retriever.
- **ORConvQA w/o first question (q) for retriever**. We do not manually include the first question of a dialog in the reformatted question for the retriever.

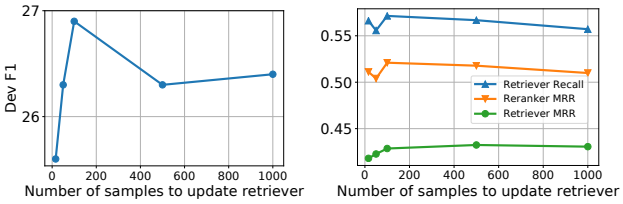
The ablation results are presented in Table 4. The following are our observations.

(1) By removing the reranker from the full system, we observe a degradation in the overall performance. Although the reranking loss does not influence the retriever, the retriever performance also decreases. This is because that the ablated system gives the best development performance earlier than the full system during training. The reason behind this is that the reader overfits before the retriever has enough fine-tuning to produce reasonable retrieval performance. This verifies our assumption that the



(a) Dev overall performance. (b) Dev retrieval performance.

Figure 4: Impact of history window size w.



(a) Dev overall performance. (b) Dev retrieval performance.

Figure 5: Impact of # samples to update retriever K_{rt}.

encoder in the reader/reranker benefits from a regularization effect by optimizing for the additional reranking task.

(2) Replacing the learned retriever with TF-IDF causes a dramatic performance drop. This further verifies our observation in Section 6.1 that a learned retriever is crucial for ORConvQA.

(3) When we do not additionally include the first question of the dialog in the reformatted question for the retriever, we observe a statistically significant performance decrease on most of the metrics. This validates our observation during data construction that the initial question of a dialog often contains a general information need that is pertinent to the entire dialog. By including the initial questions, the retriever can retrieve passages that are more relevant to the information need. The performance drop is less substantial than we anticipated. This is probably because the history window size of 6 has already covered the initial question for more than half of the questions, given that the number of history turns per question has a median of 3.

6.3 Additional Analyses

6.3.1 ***Impact of history window size.*** Leveraging conversation history is an integral part of a ConvQA system and has not been well studied in an open-retrieval setting. In this section, we study the impact of the history window size w on the system performance. The results are presented in Figure 4.

In Figure 4a, we observe that incorporating any number of history turns outperforms no history at all. Although fluctuating, the overall performance first increases then decreases, with the peak value at $w = 6$. In Figure 4b, we observe that all retrieval metrics generally grow as we incorporate more conversation history. This suggests that the additional history turns we prepend are useful for matching and retrieval in most cases. Since we have reserved 125 tokens for the reformatted question in the BERT input sequence as reported in Section 5.3, we show less degradation in the performance than previous work [33] when we prepend more history.

It is intriguing that the retriever recall, the most important retrieval metric, shows a trimodal distribution. This could be due to the "topic return" phenomenon mentioned in Yatskar [58]. Given the current question in a dialog, an adjacent turn is typically more useful than a distant turn to reveal the information need of the current turn. In other words, the utility of a history turn decreases as the distance between itself and the current turn increases. This utility trend shifts when the current turn is returning to the topic that has been discussed in a distant history turn. The trimodal distribution could imply that a topic return phenomenon typically happens five turns or nine turns away from the current turn. Moreover, the valley values of the trimodal distribution of retriever recall are consistent with those of the F1 curve in Figure 4a, suggesting that the fluctuation in the overall performance can be explained by the variation in retriever performance.

6.3.2 ***Impact of the number of passages to update retriever.*** Lee et al. [26] suggest that it is crucial to set the batch size N in the retriever pretraining phase as large as possible because it makes the pretraining task more difficult and closer to what the retriever observes at test time. During pretraining, we set N to 16 as reported in Section 5.3, meaning that we have 16 passages per question to train the retriever. At the concurrent learning phase, we can increase this number to fine-tune the question encoder in the retriever at a low cost since all passages have been encoded offline. Therefore, we investigate how helpful it is to increase the number of passages K_{rt} to fine-tune the retriever during concurrent learning. The choices of K_{rt} are [16, 50, 100, 500, 1000]. We sample the choices of K_{rt} unevenly and with large gaps so that the trends are clear. The results are presented in Figure 5.

We observe that $K_{rt} = 100$ gives the best overall performance and retriever recall. Using a smaller and larger number both give a sub-optimal performance. Although a smaller value is closer to what we use for pretraining, the retriever cannot aggressively learn from enough negative passages. On the contrary, if we use a K_{rt} value that is progressively larger than that of the pretraining time, the mismatch of supervision signals also leads to inferior performance.

7 CONCLUSIONS AND FUTURE WORK

In this work, we introduce an open-retrieval conversational QA setting as a further step towards conversational search. We create a dataset, OR-QuAC, by aggregating existing data to facilitate research on ORConvQA. We build an end-to-end system for ORConvQA, featuring a retriever, a reranker, and a reader that are all based on Transformers. Our extensive experiments on OR-QuAC demonstrate that a learnable retriever is crucial in the ORConvQA setting. We further show that our system can make a substantial improvement when we enable history modeling in all system components. Moreover, we show that the additional reranker component contributes to the model performance by providing a regularization effect. Finally, we demonstrate that the initial question of each dialog is essential for our system to understand the user's information need. For future work, we would like to address the limitations of this work by studying weak supervision methods for information-seeking conversations and a retriever that is not only learnable but

also tunable by the downstream task. In addition, we will investigate more effective history modeling methods.

ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF IIS-1715095, and in part by China Postdoctoral Science Foundation (No. 2019M652038). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

REFERENCES

[1] A. Ahmad, N. Constant, Y. Yang, and D. M. Cer. ReQA: An Evaluation for End-to-End Answer Retrieval Models. *ArXiv*, 2019.
[2] N. J. Belkin, C. Cool, A. Stein, and U. Thiel. Cases, Scripts, and Information-seeking Strategies: On the Design of Interactive Information Retrieval Systems. 1995.
[3] K. Bi, Q. Ai, Y. Zhang, and W. B. Croft. Conversational Product Search Based on Negative Feedback. In *CIKM*, 2019.
[4] D. Chen, A. Fisch, J. Weston, and A. Bordes. Reading Wikipedia to Answer Open-Domain Questions. In *ACL*, 2017.
[5] Y. Chen, L. Wu, and M. J. Zaki. GraphFlow: Exploiting Conversation Flow with Graph Neural Networks for Conversational Machine Comprehension. *ArXiv*, 2019.
[6] E. Choi, H. He, M. Iyyer, M. Yatskar, W.-T. Yih, Y. Choi, P. Liang, and L. Zettlemoyer. QuAC: Question Answering in Context. In *EMNLP*, 2018.
[7] A. Chuklin, A. Severyn, J. R. Trippas, E. Alfonseca, H. Silén, and D. Spina. Prosody Modifications for Question-Answering in Voice-Only Settings. *ArXiv*, 2018.
[8] C. Clark and M. Gardner. Simple and Effective Multi-Paragraph Reading Comprehension. In *ACL*, 2017.
[9] D. Cohen, L. Yang, and W. B. Croft. WikiPassageQA: A Benchmark Collection for Research on Non-factoid Answer Passage Retrieval. In *SIGIR*, 2018.
[10] W. B. Croft and R. H. Thompson. I3R: A New Approach to the Design of Document Retrieval Systems. *JASIS*, 38:389–404, 1987.
[11] R. Das, S. Dhuliawala, M. Zaheer, and A. McCallum. Multi-step Retriever-Reader Interaction for Scalable Open-domain Question Answering. In *ICLR*, 2019.
[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, 2019.
[13] B. Dhingra, K. Mazaitis, and W. W. Cohen. Quasar: Datasets for Question Answering by Search and Reading. *ArXiv*, 2017.
[14] M. Dunn, L. Sagun, M. Higgins, V. U. Güney, V. Cirik, and K. Cho. SearchQA: A New Q&A Dataset Augmented with Context from a Search Engine. *ArXiv*, 2017.
[15] A. Elgohary, D. Peskov, and J. L. Boyd-Graber. Can You Unpack That? Learning to Rewrite Questions-in-Context. In *EMNLP/IJCNLP*, 2019.
[16] S. Garg, T. Vu, and A. Moschitti. TANDA: Transfer and Adapt Pre-Trained Transformer Models for Answer Sentence Selection. In *AAAI*, 2020.
[17] P. M. Htut, S. R. Bowman, and K. Cho. Training a Ranking Function for Open-Domain Question Answering. In *NAACL-HLT*, 2018.
[18] H.-Y. Huang, E. Choi, and W. tau Yih. Flowqa: Grasping flow in history for conversational machine comprehension. *ArXiv*, 2018.
[19] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with GPUs. *ArXiv*, 2017.
[20] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *ACL*, 2017.
[21] V. Karpukhin, B. Ouguz, S. Min, L. Y. Wu, S. Edunov, D. Chen, and W. tau Yih. Dense Passage Retrieval for Open-Domain Question Answering. *ArXiv*, abs/2004.04906, 2020.
[22] B. Kratzwald and S. Feuerriegel. Adaptive Document Retrieval for Deep Question Answering. In *EMNLP*, 2018.
[23] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. P. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le, and S. Petrov. Natural Questions: A Benchmark for Question Answering Research. *TACL*, 7:453–466, 2019.
[24] Z.-Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *ArXiv*, 2019.
[25] J. Lee, S. Yun, H. Kim, M. Ko, and J. Kang. Ranking Paragraphs for Improving Answer Recall in Open-Domain Question Answering. In *EMNLP*, 2018.
[26] K. Lee, M.-W. Chang, and K. Toutanova. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *ACL*, 2019.
[27] R. Lowe, N. Pow, I. Serban, and J. Pineau. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *SIGDIAL*, 2015.
[28] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *ArXiv*, 2016.
[29] R. N. Oddy. *Information Retrieval through Man-Machine Dialogue.* 1977.
[30] C. Qu, L. Yang, W. B. Croft, J. R. Trippas, Y. Zhang, and M. Qiu. Analyzing and Characterizing User Intent in Information-seeking Conversations. In *SIGIR*, 2018.
[31] C. Qu, L. Yang, W. B. Croft, F. Scholer, and Y. Zhang. Answer Interaction in Non-factoid Question Answering Systems. In *CHIIR*, 2019.
[32] C. Qu, L. Yang, W. B. Croft, Y. Zhang, J. R. Trippas, and M. Qiu. User Intent Prediction in Information-seeking Conversations. In *CHIIR*, 2019.
[33] C. Qu, L. Yang, M. Qiu, W. B. Croft, Y. Zhang, and M. Iyyer. BERT with History Answer Embedding for Conversational Question Answering. In *SIGIR*, 2019.
[34] C. Qu, L. Yang, M. Qiu, Y. Zhang, C. Chen, W. B. Croft, and M. Iyyer. Attentive History Selection for Conversational Question Answering. In *CIKM*, 2019.
[35] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. SQuAD: 100, 000+ Questions for Machine Comprehension of Text. In *EMNLP*, 2016.
[36] P. Rajpurkar, R. Jia, and P. Liang. Know What You Don't Know: Unanswerable Questions for SQuAD. In *ACL*, 2018.
[37] S. Reddy, D. Chen, and C. D. Manning. CoQA: A Conversational Question Answering Challenge. *TACL*, 7:249–266, 2018.
[38] A. Shrivastava and P. Li. Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS). In *NIPS*, 2014.
[39] C. Tao, W. Wu, C. Xu, W. Hu, D. Zhao, and R. Yan. Multi-Representation Fusion Network for Multi-Turn Response Selection in Retrieval-Based Chatbots. In *WSDM*, 2019.
[40] P. Thomas, D. J. McDuff, M. Czerwinski, and N. Craswell. MISC: A data set of information-seeking conversations. In *SIGIR (CAIR'17)*, 2017.
[41] J. R. Trippas, D. Spina, L. Cavedon, and M. Sanderson. How Do People Interact in Conversational Speech-Only Search Tasks: A Preliminary Analysis. In *CHIIR*, 2017.
[42] J. R. Trippas, D. Spina, L. Cavedon, H. Joho, and M. Sanderson. Informing the Design of Spoken Conversational Search: Perspective Paper. In *CHIIR*, 2018.
[43] J. R. Trippas, D. Spina, P. Thomas, M. Sanderson, H. Joho, and L. Cavedon. Towards a Model for Spoken Conversational Search. *ArXiv*, 2019.
[44] A. Trischler, T. Wang, X. Yuan, J. Harris, A. Sordoni, P. Bachman, and K. Suleman. NewsQA: A Machine Comprehension Dataset. In *Rep4NLP@ACL*, 2016.
[45] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. In *NIPS*, 2017.
[46] E. M. Voorhees and D. M. Tice. The TREC-8 Question Answering Track Evaluation. In *TREC*, 1999.
[47] M. Wang, N. A. Smith, and T. Mitamura. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA. In *EMNLP-CoNLL*, 2007.
[48] S. Wang, M. Yu, X. Guo, Z. Wang, T. Klinger, W. Zhang, S. Chang, G. Tesauro, B. Zhou, and J. Jiang. R3: Reinforced Ranker-Reader for Open-Domain Question Answering. In *AAAI*, 2018.
[49] Y. Wu, W. Y. Wu, M. Zhou, and Z. Li. Sequential Match Network: A New Architecture for Multi-turn Response Selection in Retrieval-based Chatbots. In *ACL*, 2016.
[50] R. Yan, Y. Song, and H. Wu. Learning to Respond with Deep Neural Networks for Retrieval-Based Human-Computer Conversation System. In *SIGIR*, 2016.
[51] R. Yan, Y. Song, X. Zhou, and H. Wu. "Shall I Be Your Chat Companion?": Towards an Online Human-Computer Conversation System. In *CIKM*, 2016.
[52] L. Yang, H. Zamani, Y. Zhang, J. Guo, and W. B. Croft. Neural Matching Models for Question Retrieval and Next Question Prediction in Conversation. *ArXiv*, 2017.
[53] L. Yang, M. Qiu, C. Qu, J. Guo, Y. Zhang, W. B. Croft, J. Huang, and H. Chen. Response Ranking with Deep Matching Networks and External Knowledge in Information-seeking Conversation Systems. In *SIGIR*, 2018.
[54] L. Yang, J. Hu, M. Qiu, C. Qu, J. Gao, W. B. Croft, X. Liu, Y. Shen, and J. Liu. A Hybrid Retrieval-Generation Neural Conversation Model. In *CIKM*, 2019.
[55] L. Yang, M. Qiu, C. Qu, C. Chen, J. Guo, Y. Zhang, W. B. Croft, and H. Chen. IART: Intent-aware Response Ranking with Transformers in Information-seeking Conversation Systems. In *WWW*, 2020.
[56] W. Yang, Y. Xie, A. Lin, X. Li, L. Tan, K. Xiong, M. Li, and J. Lin. End-to-End Open-Domain Question Answering with BERTserini. In *NAACL-HLT*, 2019.
[57] Y. Yang, W.-T. Yih, and C. Meek. WikiQA: A Challenge Dataset for Open-Domain Question Answering. In *EMNLP*, 2015.
[58] M. Yatskar. A Qualitative Comparison of CoQA, SQuAD 2.0 and QuAC. In *NAACL-HLT*, 2018.
[59] Y.-T. Yeh and Y.-N. Chen. FlowDelta: Modeling Flow Information Gain in Reasoning for Conversational Machine Comprehension. *ArXiv*, 2019.
[60] Y. Zhang, X. Chen, Q. Ai, L. Yang, and W. B. Croft. Towards Conversational Search and Recommendation: System Ask, User Respond. In *CIKM*, 2018.
[61] C. Zhu, M. Zeng, and X. Huang. SDNet: Contextualized Attention-based Deep Network for Conversational Question Answering. *ArXiv*, 2018.