monoQA: Multi-Task Learning of Reranking and Answer Extraction for Open-Retrieval Conversational Question Answering

Sarawoot Kongyoung[1], Craig Macdonald[2], Iadh Ounis[2]
University of Glasgow, UK
[1]s.kongyoung.1@research.gla.ac.uk
[2]{craig.macdonald,iadh.ounis}@glasgow.ac.uk

Abstract

To address the Conversational Question Answering (ORConvQA) task, previous work has considered an effective three-stage architecture, consisting of a *retriever*, a *reranker*, and a *reader* to extract the answers. In order to effectively answer the users' questions, a number of existing approaches have applied multi-task learning, such that the same model is shared between the reranker and the reader. Such approaches also typically tackle reranking and reading as classification tasks. On the other hand, recent text generation models, such as monoT5 and UnifiedQA, have been shown to respectively yield impressive performances in passage reranking and reading. However, no prior work has combined monoT5 and UnifiedQA to share a single text generation model that directly extracts the answers for the users instead of predicting the start/end positions in a retrieved passage. In this paper, we investigate the use of Multi-Task Learning (MTL) to improve performance on the ORConvQA task by sharing the reranker and reader's learned structure in a generative model. In particular, we propose monoQA, which uses a text generation model with multi-task learning for both the reranker and reader. Our model, which is based on the T5 text generation model, is fine-tuned simultaneously for both reranking (in order to improve the precision of the top retrieved passages) and extracting the answer. Our results on the OR-QuAC and OR-CoQA datasets demonstrate the effectiveness of our proposed model, which significantly outperforms existing strong baselines with improvements ranging from +12.31% to +19.51% in MAP and from +5.70% to +23.34% in F1 on all used test sets.

1 Introduction

Research in Conversational Search and Conversational Question Answering (ConvQA) is of increasing interest. The ConvQA task (Choi et al., 2018; Reddy et al., 2019) consists in understanding the question based on a given conversational history, and extracting an answer from a given passage. This task is an extractive type of QA, meaning that the answer takes the form of a span in the provided passage, and can be successfully tackled by employing an extractive or generative *reader*. *Extractive* reader models (Qu et al., 2019a,b; Yeh and Chen, 2019) are typically employed, where the goal is to classify the start and end positions of the answer span in the given passage. In contrast, *generative* readers (Raffel et al., 2020; Khashabi et al., 2020; Lewis et al., 2020; Karpukhin et al., 2020) have demonstrated impressive results on the extractive QA tasks, where the goal is to generate tokens that are a subset of a passage. Recently, there has been more focus on retrieval as part of the ConvQA pipeline, known as Open-Retrieval Conversational Question Answering (ORConvQA). In this setting, the ORConvQA system needs to apply the ConvQA model upon passages retrieved from a large collection, given a question, before actually extracting the answer.

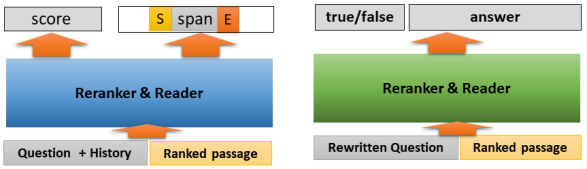

Figure 1: Overview of (a) reranker and extractive reader and (b) reranker and generative reader.

To address the ORConvQA task, prior works (Qu et al., 2020, 2021; Liang et al., 2022) have adopted a three-stage architecture, including a retriever, a reranker, and a reader to extract the answers. First, the retriever retrieves the top K relevant passages from the collection based on a question and its conversation history. The reranker and the reader then respectively rerank and identify an answer in the top K passages. We also adopt this three-stage architecture in our proposed model. However, in order to investigate the effectiveness of the

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 7207 - 7218
December 7-11, 2022

cross-encoder reranker, we consider a two-stage pipeline including a retriever and a reader, as a baseline for comparison with our system. For the retriever, existing works (Qu et al., 2020, 2021; Liang et al., 2022; Xiong et al., 2020; Yu et al., 2021) have focused on using bi-encoder dense retrieval (a question encoder and a passage encoder), which applies neural contextual language models, such as ALBERT or BERT, for encoding the question and passage into low-dimensional vectors and computing their relevance scores. For example, Yu et al. (2021) proposed ConvDR, which encodes the question and its history in a dense vector learned with a teacher-student model to mimic a dense representation of the manually rewritten question. ConvDR has also been shown to outperform other retriever models for conversational search such as sparse BM25, and bi-encoders using ALBERT (Qu et al., 2020) or BERT (Xiong et al., 2020; Karpukhin et al., 2020). Due to the good effectiveness of bi-encoder dense retrievers for passage retrieval, we adapt this type of retrieval model as our retriever. We also consider other recent existing bi-encoder passage retrievers such as TCT-ColBERT (Lin et al., 2021b, 2020a) and CQE (Lin et al., 2021a) as baseline passage retrievers.

Recently, Multi-Task Learning (MTL), which is a method of learning multiple different but related tasks at the same time, has become a popular approach for tackling several tasks using a uniform model (Qu et al., 2019b). For instance, MTL has been employed in order to efficiently answer the questions posed by the users (Qu et al., 2020, 2021). In this manner, the network structure is shared between the reranker and the reader. Doing this, existing works (Qu et al., 2020, 2021) also typically approach reranking and *extractive* reading as classification tasks, with two fully-connected layers (one for the reranker and reader, respectively) added to find an answer span for the retrieved passages (start/end positions) as well as to predict the relevance score of the question to the passage as shown in Figure 1(a). In this paper, we use the multi-task learning of the reranker and the *extractive* reader as our strongest baseline.

On the other hand, Nogueira et al. (2020) proposed monoT5, a text generation model, which was fine-tuned to generate the tokens "true" or "false" depending on whether the document is relevant or not to the query. Indeed, the monoT5 model has been shown to outperform BERT-based models in passage reranking (Nogueira et al., 2020). In addition, many studies (Raffel et al., 2020; Khashabi et al., 2020; Lewis et al., 2020; Karpukhin et al., 2020) have focused on developing a generative reader which is fine-tuned as a text generation model to extract the answer from the passage. In particular, Khashabi et al. (2020) introduced the UnifiedQA model, which has been shown to yield impressive performances on many extractive QA datasets. However, we show that compared to using monoT5 and UnifiedQA separately, a joint learning can enhance the learning efficiency and prediction accuracy of a model for the ORConvQA task, since by sharing the learning model the reranker and reader can simultaneously predict the answer and reranking score. Indeed, a joint learning by sharing a single model trained using MTL reduces the memory needs and speeds up inference (Sun et al., 2020; Standley et al., 2020). In addition, we combine the effective monoT5 (to rerank the retrieved passages) and UnifiedQA (to extract the answer from the highest scored passage) models into a strong baseline. To the best of our knowledge, no prior work has combined monoT5 and UnifiedQA by sharing a single text generation model, in order to directly extract the answers instead of predicting the start/end positions in a retrieved passage.

In summary, in this work, we investigate the use of Multi-Task Learning (MTL) to improve performance on the ORConvQA task by sharing the reranker and reader's learned structure. We propose monoQA, which uses a text generation model with multi-task learning for both the reranker and reader. Our model, which is based on the T5 (Raffel et al., 2020) text generation model, is fine-tuned simultaneously for both reranking (in order to improve the precision of the top retrieved passages) and extracting the answer. Unlike previous work, monoQA makes predictions by generating the first token for the passage reranking task, followed by the other tokens for the answer extraction task, as illustrated in Figure 1(b). Our contributions are summarised as follows: (1) we leverage Multi-Task Learning with a text generation model by sharing the reranker and reader's learned structure to effectively addresses the ORConvQA task; (2) using two different ORConvQA datasets, we compare our model to two strong baselines from the literature, and show that our MTL reranker and generative reader approach yields the best F1, Recall, MRR,

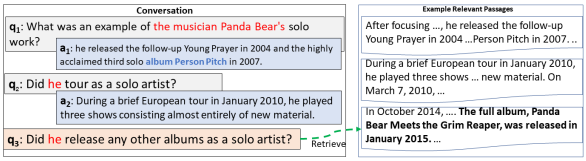

Figure 2: An example dialog and relevant passages from the ORConvQA dataset (Qu et al., 2020).

and MAP performance improvements over the strongest baseline with statistically significant improvements ranging from +5.70% to +23.34%; (3) the proposed MTL model combining the reranker and generative reader significantly outperforms and is twice as fast for inference than the individual application of the monoT5 and UnifiedQA models for reranking and extracting the answer.

2 A Three-Stage Pipeline for a ORConvQA system

We first define the ORConvQA task in Section 2.1. An overview of the proposed MTL text generation model follows in Section 2.2. Then, we explain how to fine-tune the monoQA model in Section 2.3.

2.1 Open-Retrieval Conversational Question Answering Task

Following Qu et al. (2020), the ORConvQA task is defined as follows: given a current question q_c, and a conversation history H_c – consisting of a list of $c-1$ questions and the ground truth answer pairs, i.e. $H_c = [\langle q_i, a_i \rangle]_{i=1}^{c-1}$ – the task is to predict an answer a_c for the current question q_c from a passage collection C. An example of the ORConvQA task is shown in Figure 2. It consists of a relevant passage (shown in bold), and a conversation history of length $c = 3$ (two previous pairs of questions and answers). In order to answer question q_3, the ORConvQA system needs to retrieve a list of relevant passages in C and leverage the conversation history H_c to understand and correctly predict an answer a_3 from all relevant passages in C.

2.2 Model Overview

To tackle the task described in Section 2.1, following Qu et al. (2020), our system consists of three main components: (1) a retriever for relevant passages retrieval; (2) a passage reranker for improving the precision of the top retrieved passages; and (3) a passage reader for generating the answer from the top retrieved passage. In particular, we present monoQA, which uses a text generation model with Multi-Task Learning for both the reranker and reader. First, the retriever retrieves the top K relevant passages from the collection based on a question and its history as described in Figure 3. In order to produce the final answer, the reranker and reader then re-score and identify the answer in the top-K passages from the retriever. In doing so, a single model application gives an answer to both stages - i.e., whether this is a relevant passage and the position of the answer in the passage. We now describe each component of our model in detail.

2.2.1 Retriever

The left part of Figure 3 presents the retriever component. Following Yu et al. (2021), we use a dual-encoder model named ConvDR, which consists of a question encoder and a passage encoder. The ConvDR model first encodes the current question concatenated with all previous questions and passage into the embedding space:

$$E_q = ConvDR(q_c; q_{1:c-1}),$$
$$E_p = ConvDR(p) \quad (1)$$

where q_c, $q_{1:c-1}$ and p denote the current question, the historical questions, and a passage, respectively. E_q and E_p are the embeddings of q_c concatenated with $q_{1:c-1}$ and p, respectively. ConvDR uses the dot product of E_q and E_p to calculate the retrieval score of a passage p for the current question q_c, with historical questions $q_{1:c-1}$:

$$score^{rt}(q_c; q_{1:c-1}, p) = E_q \cdot E_p \quad (2)$$

To retrieve the top K passages in the embedding space, ConvDR uses FAISS (Johnson et al., 2021), which is a library for efficient approximate nearest neighbour search. Yu et al. (2021) provides further details on ConvDR. Finally, we note that, in practice, a *rewritten* formulation q_r of the current question q_c can be used to resolve ambiguities such as coreference resolution.

2.2.2 monoQA: Reranker & Generative Reader

The right part of Figure 3 presents our proposed model, which uses T5, a large pre-trained language models designed for text generation. In particular, text generation approaches can be trained to generate a meaningful textual response based on some input text. Moreover, like BERT (Devlin et al., 2019), the pre-trained BART (Lewis et al., 2019) and T5 (Raffel et al., 2020) text generation models can be fine-tuned to perform a variety of downstream tasks. To adopt an MTL approach to a text generation model for jointly learning from both passage reranking and answer extraction, the

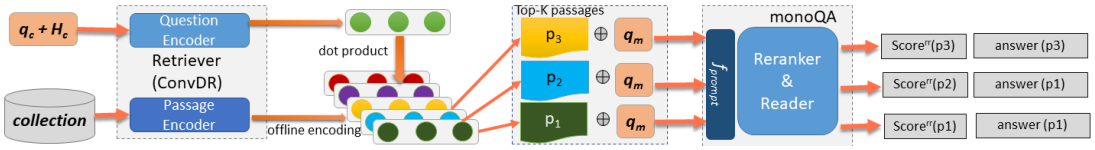

Figure 3: The overall framework of our ORConvQA system (consisting of ConvDR & monoQA).

MTL model makes predictions by generating the first token for the passage reranking task and the follow-up tokens for the answer extraction task. In particular, when fine-tuning the T5 model for a downstream task, we use Prompt Learning, which is a method to modify the model by using a task-specific prompt together with the input (Liu et al., 2021). We deploy a T5 model to capture the relation between the rewritten question q_r of the current question q_c and the passage p as shown in the right part of Figure 3. In particular, we define a monoQA transformation function as $monoQA(\cdot)$ by taking the input sequence as follows:

$$monoQA(f_{prompt}(q_r, p)) \rightarrow w_1, w_2, ..., w_n \quad (3)$$

where $f_{prompt}()$ is a prompt function (template) to format q_r, and the passage into an input sequence for monoQA. The model is then fine-tuned to generate n target tokens, as shown in Equation (3), for the token w_1 namely "true" or "false"[1] depending on whether the passage is relevant or not to question q_r, while the follow-up tokens $w_2, ..., w_n$ are the output sequence for the answer of the question q_r.

At inference time, following Nogueira et al. (2020), we apply a softmax only on the logits of the "true" and "false" tokens of the first generated token w_1 to calculate the reranker score as follows:

$$score^{rr} = \text{softmax}(w_1) \quad (4)$$

2.3 monoQA Training

Given K retrieved passages and a rewritten question q_r (see Section 2.2.1), our monoQA model jointly trains the reranker and the reader as follows:

Joint training: We consider how to fine-tune monoQA in order to generate the tokens for both passage reranking and answer extraction. In particular, the prompt function (Equation (3)) formats a question q and a passage p into an input sequence for monoQA and monoQA then outputs the contextual representation h. After that, the monoQA decoder takes the previously generated tokens as input and performs attention over h and then generates the next token. In particular, given a tuple $\langle q, p, a \rangle$, the training objective is to minimise the following loss function:

$$\mathcal{L}_{gen} = \sum_{i=1}^{M} \log P(a_i | h, a_{:i}) \quad (5)$$

where M is the number of tokens in the ground truth answer a, a_i is the i^{th} token in a, and a_0 is the beginning of sequence token (<s>).

We also consider *relevance accuracy* and word-level F1 scores for selecting the best training model checkpoint during the evaluation step with the development set of the OR-QuAC dataset. Relevance accuracy is defined as the percentage of correct predictions for the first token generated from the model. The target token is "true" (the passage is indeed relevant) or "false" (a non-relevant passage). Following Qu et al. (2020), the word-level F1 is calculated by first removing stopwords and then considering the overlapping portion of the words in the prediction and ground truth answer.

Prompt: Recently, Prompt Learning, which is a method to tailor pre-trained language models to downstream tasks by using a task-specific prompt together with the input, has recently become a popular approach for tackling several tasks using a uniform model (Liu et al., 2021). To fine-tune the monoQA model for passage reranking and answer extraction, we use Prompt Learning to modify the model input. By doing this, we investigate several prompts in previous works (Nogueira et al., 2020; Khashabi et al., 2020), and for completeness we evaluate all of them in order to choose the most effective. Details of the Prompt Learning and their corresponding experiments and results are provided in Appendix A.2. We do not investigate Prompt Learning for question reformulation since our main contribution focuses on leveraging the output of a generative model for re-ranking and reading.

Positive and negative passages: Selecting positive and negative passages is a crucial step for training monoQA. For instance, passages relevant to a question are provided in the ORConvQA task. All other passages in the collection, which are unjudged, can be viewed as non-relevant by default. To cope with this issue, following Yu et al. (2021), we employ a hard negative sampling technique by randomly selecting the negative passage p^- for the question

[1] We choose "true" and "false" as target tokens following monoT5 (Nogueira et al., 2020).

q from the top K retrieved passages by ConvDR. For training monoQA, the output sequence for the positive passage p^+ begins with the token "true" followed by the ground truth answer; the output sequence for the negative passage p^- begins with the token "false" followed by "CANNOTANSWER".

Model initialisation: We consider the use of different models to initialisation monoQA during training, since we propose to combine monoT5 and UnifiedQA to share a single text generation model. Moreover, both monoT5 and UnifiedQA are fine-tuned based on the `t5-base` model. Therefore, we investigate which of monoT5, UnifiedQA, and `t5-base`, are suitable for initialising monoQA (see details in Section 4.1).

3 Experimental Setup

Our experiments address the three following research questions: **RQ1:** Which model to use for initialising monoQA, namely which of: monoT5, UnifiedQA, and `t5-base`, lead to the best performance of monoQA on the ORConvQA task? **RQ2:** How does monoQA compare to other existing baselines, namely: (1). the ORConvQA system proposed by Qu et al. (2020); (2). the ORConvQA system proposed by Qu et al. (2020) but using ConvDR as a retriever; and (3). using ConvDR as a retriever, monoT5 as a reranker, and UnifiedQA as a reader? **RQ3:** How does our proposed system, which is a three-stage pipeline (retriever, reranker, and reader), compare to the two-stage pipeline baselines?

3.1 Datasets

To conduct our evaluation of monoQA, we choose the OR-QuAC (Qu et al., 2020) and OR-CoQA (Qu et al., 2021) datasets, which are extractive Question Answering (QA) datasets. However, the OR-CoQA dataset can be also considered as a generative QA dataset because it contains both span and freeform answers. Indeed, in this paper, we focus on extractive QA only since we train our monoQA only on the OR-QuAC training set. In addition, following (Qu et al., 2021), we remove unanswerable questions from both datasets.

OR-QuAC: This dataset has been introduced by Qu et al. (2020), adapting the well-known QuAC (Choi et al., 2018) dataset to an open-retrieval setting. This dataset is an aggregation of three existing datasets consisting of (1) the QuAC (Choi et al., 2018) dataset, which is an information seeking dataset, (2) the CANARD (Elgohary et al., 2019) dataset, which contains questions that humans have re-written from questions in the QuAC dataset, and (3) the Wikipedia corpus, a large collection of over 11 million passages, which are used as the knowledge source for actually answering a given question.

OR-CoQA: Qu et al. (2021) introduced this dataset by aggregating the CoQA (Reddy et al., 2019) dataset with the Wikipedia corpus from the OR-QuAC dataset. In contrast to OR-QuAC, the gold passages for each question are not included in the OR-CoQA dataset. Moreover, unlike OR-QuAC, there are no manually rewritten questions in the OR-CoQA dataset. As a result, we do not use OR-CoQA for training monoQA.

As exemplified in Figure 2, a question in either OR-QuAC and OR-CoQA can be ambiguous and difficult to understand without its context (e.g., q3: "Did he release any other albums as a solo artist?"). At training time, we fine-tune monoQA by using a manually rewritten query (q_r), which is provided by the OR-QuAC dataset. Thereafter, at inference time, following Dalton et al. (2020, 2021); Lin et al. (2020b), we employ another T5 model trained using the CANARD dataset to rewrite the OR-QuAC and OR-CoQA test set questions into context-independent questions that can be used as input for monoQA.

To validate our monoQA model during training, we use the OR-QuAC development set by selecting only positive examples (ground truth consisting of an answer and the corresponding passage for the question) after removing the unanswerable questions following (Qu et al., 2021). This development set consists of 490 dialogues with 2808 questions in total.

3.2 Baselines and Implementation Details

Baselines: To demonstrate the effectiveness of our proposed monoQA model, we compare it with the seven baseline systems listed as ***(a)***-***(g)*** below:

Three-stage pipelines: ***(a)*** The first ORConvQA system has been proposed by Qu et al. (2020). It adopts a duo-ALBERT encoder as the retriever and an MTL of the reranker and reader by sharing a BERT encoder. We make use of the code provided by Qu et al. (2020); ***(b)*** This baseline is adapted from (a) by replacing the duo-ALBERT encoder passage retriever with ConvDR (Yu et al., 2021) in a similar manner to our proposed

ORConvQA system (consisting of ConvDR and monoQA). This is a crucial baseline to compare with our monoQA model in order to evaluate the reranker and reader performances. We reproduce the MTL of the reranker and reader models and its evaluation results provided by Qu et al. (2020); ***(c)*** This baseline uses ConvDR as the passage retriever similarly to our ORConvQA system, monoT5 (Nogueira et al., 2020) as the passage reranker, and UnifiedQA (Khashabi et al., 2020) as the passage reader. It is deployed by using three models in the pipeline for comparison with our ORConvQA system, which employs a MTL of the reranker and reader. This comparison is done in order to evaluate the performance of using monoT5 and UnifiedQA separately in comparison with the joint learning of the reranker and reader.

Two-stage pipelines: ***(d)*** This baseline uses ConvDR as the passage retriever, and our monoQA reader as the passage reader without using the reranking results from the monoQA reranker. The reader directly identifies an answer in the top passage from the retriever; ***(e)*** This baseline uses ConvDR as the passage retriever and UnifiedQA as the passage reader. ***(f)*** This baseline uses TCT-ColBERT (Lin et al., 2020a, 2021b) as the passage retriever and UnifiedQA as the passage reader; ***(g)*** This baseline uses CQE (Lin et al., 2021a) as the passage retriever and UnifiedQA as the passage reader. The results of this baseline come from the previous work of Lin et al. (2021a).

Hyperparameter settings: Appendix A.1 describes in detail the hyper-parameter settings.

Evaluation metrics: Since we are using the OR-QuAC dataset, we naturally adopt the two evaluation metrics, namely the word-level F1, and the human equivalence score (HEQ). Word-level F1, commonly used in the Machine Comprehension and ConvQA tasks (Rajpurkar et al., 2016, 2018; Choi et al., 2018), evaluates the word overlap between the system's prediction and the ground truth answer span. Meanwhile, the HEQ metric is used to evaluate the percentage of examples for which the deployed model's F1 is equivalent to or higher than the human F1. Given n references (ground-truth answers) for each question, human F1 is calculated by averaging the maximum F1 from each $n-1$ subset with respect to the heldout reference (Choi et al., 2018). This metric is composed of HEQ-Q, computed at the question level, and HEQ-D, computed at the dialogue (conversation) level. To evaluate the retrieval performance, following (Qu et al., 2020; Yu et al., 2021), we use Mean Average Precision (MAP@10), Mean Reciprocal Rank (MRR@5) and Recall@5 as metrics for the reranker. For each query, the top 100 passages are considered. Finally, we use the McNemar's test to test statistical significance between the various readers' performances and the paired t-test for testing significant differences between the rerankers' performances.

Table 1: Effectiveness of various initialisations for training monoQA. † and ‡ denote a performance significantly worse than the model initialised using monoT5 and `t5-base`, respectively (McNemar's test, $p < 0.05$).

Model Initialisation	Retrieval			QA		
	MAP@10	Recall@5	MRR@5	F1	HEQ-Q	HEQ-D
monoT5	**0.713**	**0.804**	**0.743**	45.2	**37.2**	4.5
UnifiedQA	0.705	0.801	0.734	43.6†‡	34.9	**5.3**
`t5-base`	0.705	0.799	0.735	**45.4**	36.9	3.8

4 Experimental Results

We now address RQs 1-3 (see Section 3) and conclude with an efficiency analysis.

4.1 RQ1: Model Initialisation

In this section, we examine the effectiveness of the use of the models for initialising monoQA, namely monoT5, UnifiedQA, and `t5-base`, on the test set of OR-QuAC. All models are trained on the OR-QuAC training set by using positive passages p^+ and negative passages p^- as described in Section 2.3. In particular, we use "Question Answering: {q} [sep] {p}" as the prompt function because it performed the best according to the experiments in Appendix A.2. Table 1 presents the results for each evaluated model on the retrieval and question answering (QA) metrics.

From the table, we see that training monoQA when initialised by monoT5 achieves the highest performance on the retrieval metrics (MAP@10, Recall@5, and MRR@5). However, there are no significant differences between all of the models' retrieval performances. For the QA performance, the best word-level F1, HEQ-Q, and HEQ-D scores are obtained by the models that use `t5-base`, monoT5, and UnifiedQA, respectively. In particular, in terms of word-level F1, initialising from monoT5 or `t5-base` significantly outperforms the model trained from UnifiedQA, but both monoT5 and `t5-base` initialisations lead to comparable performances.

Therefore, in response to RQ1, we find that the model initialised from monoT5 has the best overall

Table 2: Evaluation results on OR-QuAC and OR-CoQA compared to the baselines. † denotes a performance significantly worse than our proposed monoQA model in terms of word-level F1 (McNemar's test, $p < 0.05$); ‡ denotes a performance significantly worse than our proposed monoQA model in terms of MAP@10, Recall@5, and MRR@5 (paired t-test, $p < 0.05$); The highest value for each measure is highlighted.

	Retriever	Reranker	Reader	OR-QuAC						OR-CoQA		
				Retrieval			QA			QA		
				MAP@10	Recall@5	MRR@5	F1	HEQ-Q	HEQ-D	F1	HEQ-Q	HEQ-D
	ConvDR (retriever only)	-	-	0.617	0.745	0.631	-	-	-	-	-	-
	Three-stage pipeline (retriever, reranker, and reader)											
(a)	bi-encoder (ALBERT)	BERT (MTL of reranker and extractive reader)		-	0.314‡	0.309‡	29.4†	23.7	1.3	-	-	-
(b)	ConvDR	BERT (MTL of reranker and extractive reader)		0.518‡	0.629‡	0.541‡	29.8†	22.8	2.3	28.1†	18.9	0
(c)	ConvDR	monoT5	UnifiedQA	0.590‡	0.727‡	0.618‡	22.2†	11.0	1.6	31.6†	**22.8**	0
	Two-stage pipeline (retriever and reader)											
(d)	ConvDR	-	monoQA (reader)	0.617‡	0.745‡	0.631‡	32.9‡	26.6	3.5	21.9‡	10.6	0
(e)	ConvDR	-	UnifiedQA	0.617‡	0.745‡	0.631‡	19.6‡	9.5	1.0	20.7‡	13.2	0
(f)	TCT-ColBERT	-	UnifiedQA	0.370‡	0.501‡	0.386‡	14.1‡	6.2	1.0	16.7‡	9.8	0
(g)	CQE (Lin et al., 2021a)	-	ORConvQA (reader)	-	0.415	0.310	32.0	-	-	-	-	-
(ours)	ConvDR	monoQA (MTL of reranker and generative reader)		**0.713**	**0.804**	**0.743**	**45.2**	**37.2**	**4.5**	**37.3**	19.7	0

effectiveness, yielding statistically significant improvements in word-level F1 over using UnifiedQA on the test set of the OR-QuAC dataset. In the following, we use monoT5 to initialise monoQA for answering RQ2 and comparing with the baselines.

4.2 RQ2: Effectiveness of monoQA

We investigate the performances of our monoQA model in comparison to the baselines (a)-(c) (described in Section 3.2) on the test sets of the OR-QuAC and the OR-CoQA datasets. In Table 2, the first row shows the results of ConvDR as the retriever only and the last row shows the results of our monoQA model. Table 2 (top-half) also shows the results of the existing baselines (a)-(c). In the table, on the OR-CoQA test set, we only include the question answering (QA) results since the gold passages for each question are not provided.

From the table, on the test sets of the OR-QuAC and OR-CoQA datasets, we observe that our monoQA model achieves the highest performance by significantly outperforming all baselines on all measures, excepting the baseline using monoT5 as the reranker and UnifiedQA as the reader in terms of HEQ-Q on OR-CoQA. From the table, we also observe that the baselines (b) and (c) have lower retrieval performances compared to the results of using ConvDR as the retriever only. According to these findings, the reranker of the baselines (b) and (c) might have a negative impact on the retrieval performance of the top retrieved passages. Hence, this might also lead to reducing the performances of the reader of the (b) and (c) baselines. Moreover, we further analyse why our monoQA model does not outperform the baseline (c) in terms of HEQ-Q on OR-CoQA. We find that the average number of tokens in the OR-CoQA's answer (2.6 tokens per answer) is remarkably short compared to that of the OR-QuAC's answer (14.7 tokens per answer) (Qu et al., 2021), and the predicted answer from the baseline (c) is shorter than that of our monoQA model. This prediction may lead to our monoQA model having a lower HEQ-Q score than the baseline (c). As described in Section 3.1, our proposed monoQA model is fine-tuned on the OR-QuAC dataset and evaluated on the OR-QuAC and OR-CoQA datasets. We postulate that this explains why evaluating the model with OR-CoQA exhibits a lower performance. Recall that OR-CoQA has no relevance assessments for retrieval, and hence we are unable to train a retrieval model for that dataset (which has shorter answers than OR-QuAC).

In answer to RQ2, we conclude that our proposed monoQA's joint learning of the reranker and the reader by sharing a single text generation model, does help to improve the overall performance, yielding statistically significant improvements over the baselines on both the OR-QuAC and OR-CoQA datasets. It is also of note that such a joint learning can enhance the performances of the models on the ORConvQA task compared to using monoT5 and UnifiedQA separately. Later in Section 4.4, we also analyse the efficiency of our monoQA model compared with the individual application of monoT5 and UnifiedQA.

The performances of baselines (b) and (c) on OR-QuAC raise the question as to how do the baselines (b) and (c) compare to our monoQA model when using the ground truth passages provided in the OR-QuAC test set instead of using the retrieved passages. We provide such an analysis in Appendix A.4, which shows that our monoQA reader achieves the best performance and significantly outperforms the reader of the baselines (b) and (c) on all measures.

4.3 RQ3: Effectiveness of using a Reranker

Finally, we examine the effectiveness of our proposed system, which is a three-stage pipeline using a jointly trained cross-encoder for the reranker and reader, compared to the two-stage pipeline baselines (d)-(g), which each uses a bi-encoder for retrieval as input into a reader (see details in Section 3.2). This allows to establish the impact of the reranker. Table 2 (bottom-half) presents the results of the baselines (d)-(g). From the table, we observe that our system, which is a three-stage pipeline using ConvDR as the retriever and monoQA as both the reranker and reader, achieves the highest performance by significantly outperforming all two-stage baselines on all measures – e.g. see *row (d)* vs. the last row in Table 2. In answer to RQ3, we conclude that integrating the reranker in the pipeline does help to improve both the retrieval and QA performances, yielding statistically significant improvements over the two-stage pipeline baselines.

4.4 Efficiency of monoQA

In this section, we measure the efficiency of inference of monoQA, which jointly learns the reranker and reader, in comparison with using monoT5 as the reranker and UnifiedQA as the reader separately on the test set of the OR-QuAC dataset. We find that the average prediction time of monoQA is 23ms, whereas the average prediction time of using monoT5 as the reranker and UnifiedQA as the reader separately is 44ms. This is because monoQA uses a single model application for addressing both the reranker and reader stages. Indeed, we conclude that, on the test set of OR-QuAC, our monoQA model is approximately twice as fast in inference as the individual application of monoT5 and UnifiedQA for reranking and extracting the answer.

5 Related Work

In the following, we discuss related work and position our contribution in relation to Conversational Question Answering and Multi-Task Learning.

Conversational Question Answering: This is a conversational search task, where the system needs to correctly interpret a question in the context of an ongoing conversation. Most research on conversational QA focuses on conversational response ranking tasks (Dalton et al., 2019, 2020, 2021) and extractive QA tasks (Choi et al., 2018; Reddy et al., 2019; Qu et al., 2019b; Yeh and Chen, 2019). Qu et al. (2020) were first to define the task of Open-Retrieval Conversational Question Answering (ORConvQA), where the system is required to learn to retrieve top relevant passages from a large collection before extracting answers from the passages. To address the ORConvQA task, Qu et al. (2020) proposed a three-stage pipeline: (1) a retriever, (2) a reranker, and (3) a reader. Later, Qu et al. (2021) introduced a learned weakly-supervised training approach to address the problem of accessing gold passages during the training of the model on the OR-CoQA dataset in (Qu et al., 2020). One particular problem is coarse ranking, where the reranker of Qu et al. (2020) only takes the top-ranked passages as input. This makes the whole pipeline suffers from this coarse ranking, especially for situations when the golden passages are not retrieved in the top-ranked passages. Liang et al. (2022) addressed this issue by adding a post-ranker module that can push more relevant passages to the reader. However, these works typically approach reranking and reading as classification tasks to find an answer span for retrieved passages (start/end positions). Instead, we use a text generation model with multi-task learning for both the reranker and reader in order to directly extract the answers for the users instead of predicting the start/end positions in a retrieved passage.

Multi-Task Learning (MTL) in Conversational QA: MTL methods have recently been effectively implemented in existing Conversational QA works (Kongyoung et al., 2020; Qu et al., 2019b; Xu et al., 2019; Yeh and Chen, 2019; Qu et al., 2020, 2021). However, all of the tasks in these works leverage MTL by sharing the network structure between an extractive reader and its auxiliary tasks, which are typically classification tasks, such as a yes/no question prediction or a follow-up question prediction. For example, some existing works (Qu et al., 2020, 2021) have applied MTL on reranking and answer reading by adding two fully-connected layers (one for the reranker and reader, respectively) to find an answer span for the retrieved passages (start/end positions). Instead, in this paper, we leverage MTL on reranking and answer extraction by sharing a single text generation model in order to directly extract the answers instead of predicting the start/end positions in a retrieved passage. On the other hand, Ide and Kawahara (2021) recently adopted an MTL approach that involves both classification and text generation tasks. However,

they addressed a completely different use case, consisting in detecting a user's emotion-aware response, rather than conversational QA.

6 Conclusions

We proposed the Multi-Task Learning (MTL) of passage reranking and answer extraction to share a single text generation model, so as to improve effectiveness on the Open-Retrieval Conversational Question Answering (ORConvQA) task. Our experiments on two datasets, namely the OR-QuAC and OR-CoQA datasets, showed that our proposed monoQA model has the best effectiveness on these datasets, yielding statistically significant improvements over several strong baselines from the literature, e.g. the ORConvQA system proposed by Qu et al. (2020) and the individual application of the monoT5 and UnifiedQA models. Our resulting system improves the best baseline by up to 12% MAP and 23% word-level F1 on the OR-QuAC and OR-CoQA datasets. In addition, we demonstrated that including the reranker in the pipeline helps to enhance the retrieval and QA performances, yielding statistically significant improvements over the state-of-the-art two-stage pipeline baselines. Furthermore, we showed that our MTL-based monoQA model improves the efficiency of inference compared to the individual application of the monoT5 and UnifiedQA models for separately reranking and extracting the answer to the user's question.

Limitations and Future Work

Our work shows the effectiveness of the multi-task learning of the reranker and a generative reader by sharing a single text generation model on the ORConvQA task. Indeed, while our generative model is trained to generate only word sequences appearing in the input passage, we observe that 1.5% of the generated tokens are not extracted from the input. While this may not affect user satisfaction, the extractive evaluation measures may underestimate the model's utility. For this reason, it is also worth investigating the multi-task learning of the reranker and an extractive reader by sharing a single model. Another limitation of our work is that the input of monoQA is a rewritten question from another T5 model (see Section 3.1). Ideally, we would like a single model to be able to use the original question (without needing it to be first rewritten). We leave both these investigations to future work.

References

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 *Proc. EMNLP.*, page 2174–2184.

Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. 2019. TREC CAsT 2019: The conversational assistance track overview. In *Proc. TREC*.

Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. 2020. CAsT 2020: The conversational assistance track overview. In *Proc. TREC*.

Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. 2021. CAsT 2021: The conversational assistance track overview. In *Proc. TREC*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL*, pages 4171–4186.

Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. 2019. Can you unpack that? Learning to rewrite questions-in-context. In *Proc. EMNLP*, pages 5918–5924.

Tatsuya Ide and Daisuke Kawahara. 2021. Multi-task learning of generation and classification for emotion-aware dialogue response generation. In *Proc. NAACL*, pages 119–125.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with GPUs. *In Proc. IEEE Transactions on Big Data*, 7(3):535–547.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proc EMNLP*, pages 6769–6781.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UnifiedQA: Crossing format boundaries with a single QA system. In *Proc. EMNLP*, pages 1896–1907.

Sarawoot Kongyoung, Craig Macdonald, and Iadh Ounis. 2020. Multi-task learning using dynamic task weighting for conversational question answering. In *Proc. SCAI*, pages 17–26.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proc. ACL*, pages 7871–7880.

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. *In Proc. NeurIPS*, pages 9459–9474.

Tingting Liang, Yixuan Jiang, Congying Xia, Ziqiang Zhao, Yuyu Yin, and Philip S Yu. 2022. Multifaceted improvements for conversational open-domain question answering. *arXiv preprint arXiv:2204.00266*.

Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2020a. Distilling dense representations for ranking using tightly-coupled teachers. *arXiv preprint arXiv:2010.11386*.

Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021a. Contextualized query embeddings for conversational search. In *Proc. EMNLP*, pages 1004–1015.

Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021b. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *Proc. RepL4NLP*, pages 163–173.

Sheng-Chieh Lin, Jheng-Hong Yang, Rodrigo Nogueira, Ming-Feng Tsai, Chuan-Ju Wang, and Jimmy Lin. 2020b. Conversational question reformulation via sequence-to-sequence architectures and pretrained language models. *arXiv preprint arXiv:2004.01909*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. In *Proc. EMNLP*, pages 708–718.

Chen Qu, Liu Yang, Cen Chen, W Bruce Croft, Kalpesh Krishna, and Mohit Iyyer. 2021. Weakly-supervised open-retrieval conversational question answering. In *Proc. ECIR*, page 529–543.

Chen Qu, Liu Yang, Cen Chen, Minghui Qiu, W. Bruce Croft, and Mohit Iyyer. 2020. Open-retrieval conversational question answering. In *Proc. SIGIR*, pages 539–548.

Chen Qu, Liu Yang, Minghui Qiu, W. Bruce Croft, Yongfeng Zhang, and Mohit Iyyer. 2019a. BERT with history answer embedding for conversational question answering. In *Proc. SIGIR*, pages 1133–1136.

Chen Qu, Liu Yang, Minghui Qiu, Yongfeng Zhang, Cen Chen, W Bruce Croft, and Mohit Iyyer. 2019b. Attentive history selection for conversational question answering. In *Proc. CIKM*, pages 1391–1400.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. In *Proc. JMLR*, pages 1–67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proc. ACL*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proc. EMNLP*, page 2383–2392.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. CoQA: A conversational question answering challenge. In *Proc. ACL*, pages 249–266.

Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. 2020. Which tasks should be learned together in multi-task learning? In *Proc. PMLR*, pages 9120–9132.

Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. 2020. Adashare: Learning what to share for efficient deep multi-task learning. *In Proc. NeurIPS*, 33:8728–8740.

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

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *Proc. ICLR*.

Yichong Xu, Xiaodong Liu, Yelong Shen, Jingjing Liu, and Jianfeng Gao. 2019. Multi-task learning with sample re-weighting for machine reading comprehension. In *Proc. NAACL-HLT*, pages 2644–2655.

Yi-Ting Yeh and Yun-Nung Chen. 2019. FlowDelta: Modeling flow information gain in reasoning for conversational machine comprehension. In *Proc. MRQA*, pages 86–90.

Shi Yu, Zhenghao Liu, Chenyan Xiong, Tao Feng, and Zhiyuan Liu. 2021. Few-shot conversational dense retrieval. In *Proc. SIGIR*, pages 829–838.

A Appendices

Our code and data are publicly available at the following URL: `https://github.com/terrierteam/monoQA`.

Table 3: Effectiveness of various prompts for training monoQA.

Prompt	Retrieval			QA		
	MAP@10	Recall@5	MRR@5	F1	HEQ-Q	HEQ-D
monoT5	0.708	0.801	0.739	**45.6**	36.8	4.2
UnifiedQA	0.705	0.800	0.735	45.2	35.3	3.6
Our prompt.	**0.713**	**0.804**	**0.743**	45.2	**37.2**	**4.5**

A.1 Hyperparameter Settings

For ConvDR, we reproduce the model and its evaluation results provided by Yu et al. (2021) to generate the offline passage embeddings from the passage collection of the OR-QuAC dataset as shown in Figure 3. We implement the monoQA model using the following PyTorch models from HuggingFace (Wolf et al., 2020), namely `t5-base`, `castorini/monot5-base-msmarco`, and `allenai/unifiedqa-t5-base`. Following Qu et al. (2020), these models are configured as follows: the maximum sequence length is set to 512, the number of training epochs is set to 10, the batch size is set to 16, and the learning rate is set to $5e^{-5}$. The models are trained on a NVIDIA RTX A6000. The average training time of monoQA is 6.3 hours. The number of parameters in monoQA is approximately 222 million parameters, i.e. the same as monoT5 and other fine-tuned versions of `t5-base`. We save the checkpoints every epoch and evaluate on the development set of the OR-QuAC dataset. We provide the details of how to select the best checkpoint in Appendix A.3.

A.2 Prompt Learning

Recently, Prompt Learning, which is a method to modify pre-trained language models to downstream tasks by using a task-specific prompt together with the input, has increasingly become a popular approach for tackling several tasks in a uniform model (Liu et al., 2021). To fine-tune the monoQA model for passage reranking and answer extraction, we adopt Prompt Learning to modify the model input. We have observed that several prompts have previously been used in previous work (Nogueira et al., 2020; Khashabi et al., 2020). Below we list the templates $f_{prompt}()$ that we consider in this study:

• *monoT5 prompt:* We adapt the monoT5's template by replacing the prefix word from "Query:" to "Question:", the separator token from "Document:" to "Passage:", without using the word "Relevant:":

$$\text{“}Question : \{q\}\ \ Passage : \{p\}\text{”} \quad (6)$$

• *UnifiedQA prompt:* UnifiedQA (Khashabi et al., 2020) made use of a '\n' between the current question and the passage:

$$\text{“}\{q\}\ \backslash n\ \{p\}\text{”} \quad (7)$$

However, under the standard T5 tokenizer, a whitespace such as '\n' does not result in a separate token, so the end result of this formulation is a simple concatenation of the question and passage:

$$\text{“}\{q\}\ \{p\}\text{”} \quad (8)$$

• *Our prompt:* For comparison with the above templates from the literature, we design a new template using "Question Answering:" as a prefix and a T5-provided special tokens ($[sep]$) as a separator token between the question and the passage.

$$\text{“}QuestionAnswering : \{q\}\ [sep]\ \{p\}\text{”} \quad (9)$$

Table 3 shows the results of each evaluated model for each of the above prompts. From the table, we see that the monoQA model trained by using our designed prompt (Question Answering: {q} [sep] {p}) has the highest performance on all measures, except when it uses the monoT5 prompt (Question: {q} Passage: {p}) for word-level F1. To conclude, we find that the model learned with our designed prompt has the best overall effectiveness. As a consequence, we use "Question Answering: {q} [sep] {p}" as the prompt for training monoQA.

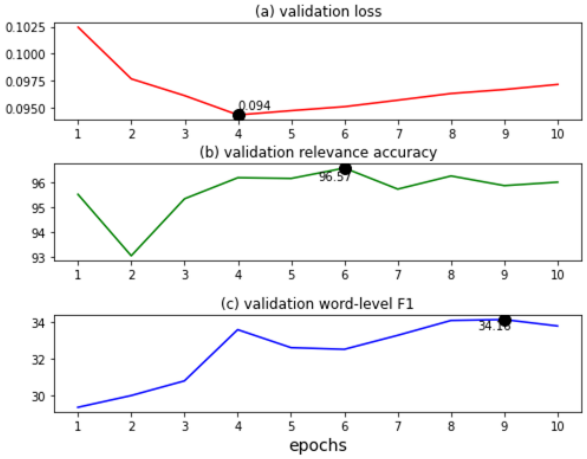

Figure 4: The validation scores of (a) loss, (b) relevance accuracy, and (c) word-level F1, for each validation step (epochs). • denotes the best epoch of each score. The best epochs of the model on loss, relevance accuracy, and word-level F1 scores, are 4, 6, and 9, respectively.

A.3 Selecting the Best Model

We further investigate how to identify the optimal training checkpoint for our proposed monoQA model on the OR-QuAC development set. The model is trained on the OR-QuAC training set by using the positive p^+ and negative passages p^- described in Section 2.3. In particular, we use "Question Answering: {q} [sep] {p}" as the prompt function and monoT5 to initialise monoQA. In this appendix, we consider the performance of each model checkpoint on both reranking and answer extraction.

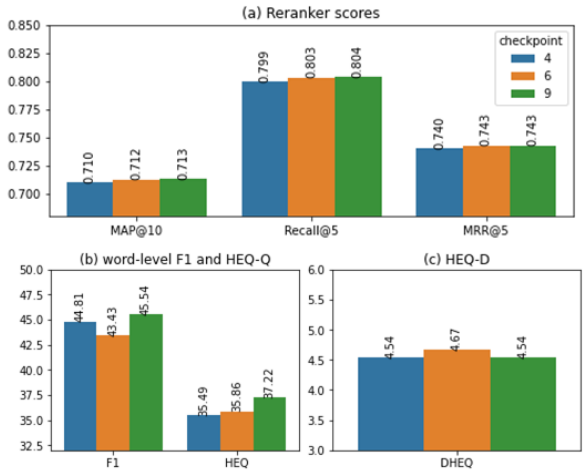

Figure 5: Results on the test set of the OR-QuAC dataset in terms of (a) MAP@10, Recall@5, and MRR@5, (b) word-level F1 and HEQ-Q, and (c) HEQ-D, of the models at epochs 4, 6, and 9.

Table 4: Evaluation results on OR-QuAC in comparison to the baselines by extracting the answer on the *ground truth passage*. † denotes a performance significantly worse than our proposed monoQA model in terms of word-level F1 (McNemar's test, $p < 0.05$). The highest value for each measure is highlighted.

Reranker	Reader	F1	HEQ-Q	HEQ-D
(b) BERT (MTL of reranker and extractive reader)		40.0‡	33.0	3.5
(c) monoT5	UnifiedQA	30.0‡	16.5	1.9
(ours) monoQA (MTL of reranker and generative reader)		**56.5**	**48.7**	**7.1**

We identify the best checkpoint of the model for each measure, namely validation loss, validation relevance accuracy, and word-level F1 as discussed in Section 2.3. Figure 4 shows the best epochs of the model in terms of validation loss, relevance accuracy, and word-level F1 scores, which are 4, 6, and 9, respectively. We then evaluate the models obtained at these epochs (4, 6, and 9) on the OR-QuAC test set, as depicted in Figure 5. Figure 5 shows that the model checkpoint at epoch 9 has the best performance in terms of MAP@10, Recall@5, MRR@5, word-level F1, and HEQ-Q, whereas in HEQ-D the epoch 6 is the best. Indeed, the model that exhibits the highest word-level F1 on the validation set is also the best model when evaluated on the test set in terms of MAP@10, Recall@5, MRR@5, word-level F1, and HEQ-Q.

A.4 Effect of Providing Ground Truth Passages

In this section, we experiment to answer the question concerning how do baselines (b) and (c) (listed in Section 3.2) compare to our monoQA model when using the ground truth passages provided in the OR-QuAC test set instead of using the retrieved passages. In particular, recall that baseline (b) is the MTL of the reranker and reader by sharing a BERT encoder, while (c) is the individual application of monoT5 and UnifiedQA. By doing this comparison, we can control the impact of the reranker, and consider only the effectiveness of the reader. Indeed, in this setting, all models predict the answer by using the question and the ground truth passage. Table 4 shows the results of each evaluated model on the test set of OR-QuAC.

On analysing Table 4, we observe that our proposed monoQA model achieves the best performance across all measures and significantly outperforms the baselines (b) and (c) in terms of word-level F1 scores according to the McNemar's test ($p < 0.05$). Indeed our monoQA model's joint learning of the reader and the reranker can indeed help improve the performance of the answer extraction.

A.5 Reproducibility Criteria

Table 5 summarises our answers to the EMNLP reproducibility criteria questions.

Table 5: Summary of Reproducibility Criteria.

For all reported experimental results:	
A clear description of the mathematical setting, algorithm, and/or model	Section 2
Link to source code	https://github.com/terrierteam/monoQA
Description of computing infrastructure used	NVIDIA RTX A6000 GPU
The average runtime for each model or algorithm (e.g., training, inference, etc.)	Appendix A.1, Section 4.4
Number of parameters in each model	Appendix A.1
Corresponding validation performance for each reported test result	Appendix A.3
Explanation of evaluation metrics used	Section 3.2
For all experiments with hyperparameter search:	
The exact number of training and evaluation runs	Appendix A.1
Bounds for each hyperparameter	Number of epochs: 1–10
Hyperparameter configurations for best-performing models	Appendix A.1
Number of hyperparameter search trials	1
The method of choosing hyperparameter values	Appendix A.3
For all datasets used:	
Relevant details such as languages, and number of examples and label distributions	Section 3.1
Details of train/validation/test splits	Section 3.1
Explanation of any data that were excluded, and all preprocessing steps	Section 3.1
Link to downloadable versions of the data	https://github.com/terrierteam/monoQA