HISTORY MODELING FOR CONVERSATIONAL INFORMATION RETRIEVAL

A Dissertation Presented

by

CHEN QU

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2021

College of Information and Computer Sciences

HISTORY MODELING FOR CONVERSATIONAL INFORMATION RETRIEVAL

A Dissertation Presented

by

CHEN QU

Approved as to style and content by:

W. Bruce Croft, Chair

Mohit Iyyer, Member

James Allan, Member

Rajesh Bhatt, Member

James Allan, Chair of the Faculty
College of Information and Computer Sciences

To My Family

ACKNOWLEDGMENTS

I would like to take a moment to express my sincere gratitude to those who guided me, helped me, and supported me throughout this wonderful journey.

First, I am deeply grateful to my advisor, Professor W. Bruce Croft, who has been offering me visionary guidance, immense support, constructive feedback, and rigorous training throughout my PhD program. My transfer from the MS program to the MS/PhD program would never have been possible without his support. Bruce's vision in information retrieval navigates me to work on research directions with a long-lasting impact. I also greatly appreciate the autonomy and flexibility allowed by Bruce during my study. The lessons I learned from Bruce had, and will continue to have, a tremendous influence on my future career.

I would like to extend my most sincere thanks to my committee members: Professors Mohit Iyyer, James Allan, and Rajesh Bhatt. Their valuable feedback has made this dissertation in better shape. In particular, I thank James for his detailed and insightful comments and suggestions on the materials presented in this dissertation. I must also thank Mohit for his advice and guidance for many core projects in this dissertation. Mohit's unique insight from an NLP perspective plays an important role in my synthesis project and this dissertation.

I would like to thank all my mentors and collaborators who have helped me to become a better researcher. I would like to especially thank Liu Yang, whose mentorship has been indispensable ever since I started at UMass (thanks Bruce again for introducing me to him!). I am super fortunate to continue having him on my side as I wrapping up the academic chapter of my life and moving to an industrial position. I would also like to thank my long-term collaborators, including Minghui Qiu,

Yongfeng Zhang, Cen Chen, Johanne R. Trippas, who have contributed to many of the research projects in this dissertation and provided profound discussions to facilitate our research. In addition, I would like to thank my mentors and colleagues from my internships, including Feng Ji, Minghui Qiu, Zhiyu Min, Haiqing Chen, Jun Huang, Chenyan Xiong, Paul Bennet, Yizhe Zhang, Corby Rosset, Weize Kong, Liu Yang, Mingyang Zhang, Michael Bendersky, and Marc Najork, for my invaluable industrial experience made possible by them. Then, I would like to thank my other co-authors and collaborators, including Kalpesh Krishna, Xinjing Huang, and Professors Hamed Zamani, Eric Learned-Miller, and Falk Scholer, for their help and collaboration at different stages of my PhD study. Finally, I would like to offer special thanks to Dr. Kan Xu and Professor Yuan Lin, who introduced me to the field of information retrieval during my undergraduate study. It was a real privilege to have this opportunity to work with all of my collaborators. My research career would have been a lot more difficult and different without them.

It was a great honor to be part of the Center for Intelligent Information Retrieval (CIIR) at UMass Amherst. I would like to thank our staff at our lab. Thanks to Dan Parker for keeping our servers running and for offering the most timely and pertinent support for my technical difficulties. Thanks to Kate Moruzzi, Jean Joyce, Glenn Stowell, and Stephen Harding for their dedicated administrative and technical support. I would also like to thank all my labmates for creating such an inclusive and collaborative working environment: Ali, Dan, Hamed, Helia, Jiepu, John, Keping, Lakshmi, Liu, Myung-ha, Negin, Puxuan, Qingyao, Rab, Shahrzad, Sheikh, Shiva, Yen-Chieh, Youngwoo, and Zhiqi. Also, I thank our Graduate Programs Managers and Assistants Eileen Hamel, Malaika Ross, and Leeanne M. Leclerc for keeping everything sorted out and for looking out for us.

Finally, I would like to conclude this chapter by thanking my family and friends. I am deeply grateful to my parents for their unwavering belief and support in me when I

decided to start this journey and for supporting me during my most vulnerable times. Thanks to my friends, who I cannot enumerate here, for creating happy distractions for me outside of my research. I am super grateful to have them in my life.

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF IIS-1715095. 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.

ABSTRACT

HISTORY MODELING FOR CONVERSATIONAL INFORMATION RETRIEVAL

SEPTEMBER 2021

CHEN QU

B.Eng., DALIAN UNIVERSITY OF TECHNOLOGY

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor W. Bruce Croft

Conversational search is an embodiment of an iterative and interactive approach to information retrieval (IR) that has been studied for decades. Due to the recent rise of intelligent personal assistants, 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. One of the major challenges of conversational search is to leverage the conversation history to understand and fulfill the users' information needs. In this dissertation work, we investigate history modeling approaches for conversational information retrieval. We start from history modeling for user intent prediction. We analyze information-seeking conversations by user intent distribution, co-occurrence, and flow patterns, followed by a study of user intent prediction in an information-seeking setting with both feature-based methods and deep learning methods. We then move to history modeling for conversational

question answering (ConvQA), which can be considered as a simplified setting of conversational search. We first propose a positional history answer embedding (PosHAE) method to seamlessly integrate conversation history into a ConvQA model based on BERT. We then build upon this method and design a history attention mechanism (HAM) to conduct a "soft selection" for conversation history. After this, we extend the previous ConvQA task to an open-retrieval (ORConvQA) setting to emphasize the fundamental role of retrieval in conversational search. In this setting, we learn to retrieve evidence from a large collection before extracting answers. We build an end-to-end system for ORConvQA, featuring a learnable dense retriever. We conduct experiments with both fully-supervised and weakly-supervised approaches to tackle the training challenges of ORConvQA. Finally, we study history modeling for conversational re-ranking. Given a history of user feedback behaviors, such as issuing a query, clicking a document, and skipping a document, we propose to introduce behavior awareness to a neural ranker. Our experimental results show that the history modeling approaches proposed in this dissertation can effectively improve the performance of different conversation tasks and provide new insights into conversational information retrieval.

CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Conversational search is an embodiment of an iterative and interactive approach to information retrieval (IR) that has been studied for decades (Belkin et al., 1995; Croft and Thompson, 1987; Oddy, 1977). Due to the recent rise of intelligent personal assistant systems, such as Siri, Alexa, AliMe, Cortana, and Google Assistant, a growing part of the population are relying on these systems to finish everyday tasks. Examples of these tasks include setting a timer or placing an order. Some users also interact with them for entertainment or even as an emotional companion. Although current personal assistant systems are capable of completing tasks and even conducting informal conversations, they cannot handle *information-seeking conversations* with complex information needs that require multiple turns of interaction. Conversational personal assistant systems serve as an appropriate media for interactive information retrieval, but much work needs to be done to enable functional conversational search via such systems. In this dissertation, we focus on the history modeling aspect of conversational IR, as a fundamental step towards conversational search.

A typical conversational search process involves multiple "cycles". In each cycle, a user first specifies an information need and then an agent (a system) retrieves answers iteratively either based on the user's feedback or by asking for missing information proactively (Zhang et al., 2018; Aliannejadi et al., 2019). The user could ask a follow-up question and shift to a new but related information need, entering the next cycle of conversational search. For example, in Table 1.1, we show that questions in an

Table 1.1: An example of an information-seeking conversation from the QuAC collection (Choi et al., 2019). Co-references and related terms are marked in the same color across history turns. Q_2, Q_4, Q_5 and Q_6 are closely related to their immediate previous turn(s) while Q_7 is related to a remote question Q_1. Also, Q_3 does not follow up on Q_2 but shifts to a new topic.

Topic: Lorrie Morgan's music career			
#	ID	Role	Utterance
1	Q_1	User	What is relevant about Lorrie's musical career?
	A_1	Agent	... her first album on that label, Leave the Light On, was released in 1989.
2	Q_2	User	What songs are included in the album?
	A_2	Agent	CANNOTANSWER
3	Q_3	User	Are there any other interesting aspects about this article?
	A_3	Agent	made her first appearance on the Grand Ole Opry at age 13,
4	Q_4	User	What did she do after her first appearance?
	A_4	Agent	... she took over his band at age 16 and began leading the group ...
5	Q_5	User	What important work did she do with the band?
	A_5	Agent	leading the group through various club gigs.
6	Q_6	User	What songs did she played with the group?
	A_6	Agent	CANNOTANSWER
7	Q_7	User	What are other interesting aspects of her musical career?
	A_6	Agent	*To be predicted ...*

information-seeking conversation could be closely related to their conversation history. It is crucial for a conversational search system to model the conversation history in order to understand and fulfill the users' information needs.

Specifically, we work on history modeling for user intent prediction, conversational question answering, open-retrieval conversational question answering, and conversational re-ranking. We present our motivations and contributions for each topic as follows.

1.1 User Intent Prediction

In an information-seeking conversation, users have multiple rounds of information exchange with conversational assistants to retrieve or specify answers. One of the challenges in this process is the difficulty of modeling the conversation about the

information need both before and after an answer has been given. An important step in modeling conversational interactions is to accurately detect and predict user intent in information-seeking conversations.

For example, a user issues the following utterance in an information-seeking conversation when trying to resolve an issue with Microsoft Office: "*After modified the Windows entry, value of regedit, the error also happened. When I use C++ for creating another new Microsoft::Office::Interop::PowerPoint::Application instance, the COMException is throwed*". In this case, the conversational assistant should be capable of first recognizing the *Negative Feedback* from the user, which suggests the last answer given by the agent is not satisfactory. We then expect the assistant to improve the previous answer by considering the *Further Details* provided by the user. Accurately detecting and predicting user intent is the fundamental starting point for a conversational assistant to process user's utterances accordingly. It is crucial to consider conversation history in the prediction process given the interactive nature of information-seeking conversations. Thus, we study history modeling for user intent prediction with the MSDialog dataset we create. MSDialog has over 2,000 information-seeking dialogs with 10,000 utterances. It is labeled with crowdsourcing with 12 user intent types as shown in Table 3.2.

1.2 Conversational Question Answering

It is natural for people to seek information through conversations. In a typical use case of conversational search, a user initiates a conversation with a specific information need. The search system conducts multiple turns of question answering (QA) interaction with the user, including asking proactively, to better understand this information need. The system then tries to fulfill this need by retrieving answers iteratively based on the user's feedback or clarifying questions. The user sometimes asks follow-up questions with a related but new information need and thus enters the

next "cycle" of the conversational search process. In order to keep track of the user's latest information need, the system should be capable to make use of the conversation history. For example, we show that conversation history is highly informative by highlighting common co-references across history turns in Table 1.1.

In our view, conversational question answering (ConvQA) can be considered as a simplified setting of conversational search, since current ConvQA systems do not focus on asking proactively. However, ConvQA is a tangible task for researchers to work on modeling the change of information needs across cycles. In this dissertation, we follow Choi et al. (2019) and Reddy et al. (2019) to formulate the ConvQA task as conversational machine comprehension, where we answer questions in a conversation by predicting an answer span in a given passage. We study history modeling under this setting as an integral part of conversational search.

1.3 Open-retrieval Conversational Question Answering

A significant limitation of the current ConvQA setting is that an answer is either selected from a *given* candidate set (Yang et al., 2018) or extracted from a *given* passage (Choi et al., 2019). This simplification neglects the fundamental role of retrieval in conversational search. To address this issue, we continue studying the ConvQA task and extending it to an open-retrieval (ORConvQA) setting, where we *learn to retrieve evidence from a large collection* before extracting answers.

The open-retrieval setting presents challenges to training the ConvQA system. We study both full supervision and weak supervision approaches for training. In the fully-supervised approach, we encourage the model to find the gold passage that comes with the dataset for a given question and extract an answer from it. In the weakly-supervised approach, we investigate the performance of span-match weak supervision and our proposed learned weak supervision. The former identifies weak answers in the retrieval results by finding a span that is an exact match to the known answer

while the latter seeks to find a paraphrased span of the known answer in a retrieved passage as the weak answer. The learned weak supervision enjoys more flexibility and can be more suitable for the long and freeform answers in information-seeking conversations.

1.4 Conversational Re-ranking

Although conversational search and ad-hoc retrieval vary greatly in many aspects, they share the same backbone of document ranking, which is the core of modern search engines. To fulfill a complicated information need with a search engine, users typically need to conduct searches for multiple turns. In each turn, the user issues or reformulates a query, browses search engine result pages (SERPs), and clicks on one or more documents for further investigation. This iterative information-seeking process bears a strong resemblance to conversational search. Different types of historical user behaviors can provide different clues of the information need. This part of the dissertation focuses on the conversational document ranking task. We are given history user behaviors, including queries, clicked documents, and skipped documents in the session, the task is to re-rank a set of candidate documents for the current query. This task deals with the challenge of user interaction modeling for conversational search via a concrete task of document ranking.

1.5 Contributions

We highlight the contributions of this dissertation as follows.

- In terms of user intent prediction, we create a large-scale annotated dataset for multi-turn information-seeking conversations, which is the first of its kind to the best of our knowledge. We perform in-depth data analysis and characterization of user intent with these conversations. We then use both feature-based machine learning methods and neural approaches for user intent prediction.

Moreover, neural models achieve significant improvements after incorporating history information.

- In terms of conversational question answering, we introduce a positional history answer embedding method to incorporate conversation history into a BERT-based machine comprehension model. We then propose a history attention mechanism to conduct a "soft selection" for conversation history turns. We show that conversation history plays a vital role in ConvQA. To learn more generalizable representations, we jointly learn answer span prediction and dialog act prediction in a multi-task learning (MTL) setting.

- In terms of open-retrieval conversational QA, we first demonstrate the importance of a learnable retriever in this task. We then show that our system can make a substantial improvement when we enable history modeling in all system components. As for supervision approaches, we verify that full supervision is more effective than weak supervision. We further show that a learned weak supervisor can outperform the span-match weak supervisor, proving the capability of learned weak supervision in dealing with freeform answers, i.e., human-generated answers that are not necessarily strict spans of any passage. Moreover, it leads to a significant improvement when we combine the learned weak supervisor with the span-match weak supervisor, indicating these two methods can complement each other.

- In terms of conversational re-ranking, we incorporate history user behaviors into a BERT-based ranker with Hierarchical Behavior Aware Transformers. We first show that a BERT ranker without any history information is able to outperform a recent context aware recurrent model (Ahmad et al., 2019b). We further show that BERT is capable of modeling session history by simply prepending history user behaviors to the current query. Moreover, we demonstrate that our

hierarchical behavior attention mechanism is significantly more powerful in this scenario than a simple concatenation. This indicates that behavior awareness is essential in conversational document ranking.

1.6 Outline

The rest of this dissertation is organized as follows. In Chapter 2, we present the background and related work of history modeling in conversational IR. In Chapter 3, we conduct user intent analysis and prediction in information-seeking conversations and study neural approaches to history modeling under this setting. In Chapter 4, we introduce history modeling methods for conversational question answering. In Chapter 5, we tackle the open-retrieval problem in conversational QA and study history modeling and training approaches for this task. In Chapter 6, we focus on the ranking aspect of conversational search and investigate history modeling approaches with the conversational ranking task. Finally, in Chapter 7, we summarize this dissertation and discuss future work in history modeling for conversational IR.

CHAPTER 2

BACKGROUND AND RELATED WORK

This dissertation is related to several research areas, including utterance intent modeling, single-turn question answering, conversational question answering, session-based retrieval, and other directions in conversational search.

2.1 Utterance Intent Modeling

Utterance intent and dialog act modeling has been studied in both information retrieval and natural language processing (NLP) domains. Stolcke et al. (2000) study dialog act prediction using hidden Markov models on the Switchboard corpus that contains spontaneous human-to-human telephone speech. Surendran and Levow (2006) adopt a combination of linear SVM with hidden Markov models and conduct dialog act tagging on the HCRC MapTask corpus (Carletta et al., 1997) that consists of instruction-like conversations. Olney et al. (2003) classify student utterances in an auto tutoring system with part of speech tagging, cascaded finite state transducers, and disambiguation rules. Bhatia et al. (2012, 2014) focus on the problem of classifying forum posts based on the purpose in the discussion thread and further apply the dialog act information to summarize the ongoing discussion. Shiga et al. (2017) work on detecting conversational information needs in spoken utterances in collaborative search tasks using temporal, dialog, semantic, linguistic, and statistical features.

Recent advances in deep learning has made it possible to conduct sentence classification on both the word level (Kim, 2014) and the character level (Zhang et al., 2015). These deep learning techniques have been applied in user intent classification

in dialog utterances. Datta et al. (2016) combine convolutional neural networks and recurrent neural networks to construct the utterance classification model and enhance it with domain-specific word embeddings. Yu et al. (2019) recognize that an utterance could serve multiple functions and apply deep learning techniques to tag an utterance with multiple dialog acts. In Chapter 3, we focus on user intent prediction and history modeling in information-seeking conversations. This specific utterance classification task presents unique challenges because that the interactive nature of information-seeking conversations requires a model to consider conversation history. The user intent information predicted by our model has been shown to be useful in facilitating response ranking in information-seeking conversations via an intent-aware utterance attention mechanism (Yang et al., 2020a).

2.2 Single-Turn Question Answering

One of the first modern reformulations of the QA task dates back to the TREC-8 Question Answering Track (Voorhees and Tice, 1999). Its goal is to answer 200 fact-based, short-answer questions by leveraging a large collection of documents. Many popular QA tasks and models either follow an answer selection setting or a machine comprehension setting. We discuss related work in both settings.

2.2.1 Answer Selection

The goal of the answer selection task is to find a piece of text, either short or long, in a candidate set that answers a given question. Wang et al. (2007) adopt a generative approach for the answer sentence selection task to model how the question can be generated from an answer via syntactic and semantic transformations. Iyyer et al. (2014) introduce a dependency tree recursive neural network to combine clues across different sentences in relatively long questions. Yang et al. (2015) create the WikiQA dataset that contains challenging and possibly unanswerable questions and

conduct baseline experiments with word-match, feature-based, and neural approaches. Yang et al. (2016) propose an attention-based neural matching model to rank short answer texts. It introduces a value-shared weighting scheme that is tailored for the semantic match between a question and an answer. Cohen and Croft (2016) adopt a Bidirectional Long Short Term Memory (BiLSTM) network with a rank sensitive loss function and focus specifically on non-factoid QA.

2.2.2 Machine Comprehension

The machine comprehension (MC) task aims to extract an answer span or synthesize an answer for a question using a given passage. It is particularly relevant to this dissertation because we adopt this technique in Chapters 4 and 5. One of the most influential benchmarks in this field is SQuAD (The Stanford Question Answering Dataset) (Rajpurkar et al., 2016, 2018). The reading comprehension task in SQuAD is conducted in a single-turn QA manner. The system is given a passage and a question. The goal is to answer the question by predicting an answer span in the passage. Extractive answers in this task enable easy and fair evaluations compared with other datasets that have abstractive answers generated by human. Other widely-used datasets include MS MARCO (Nguyen et al., 2016) that features real user search queries and human rewritten answers, TriviaQA (Joshi et al., 2017) that features complex and compositional questions, and Google Natural Questions (Kwiatkowski et al., 2019) that features real user questions with both long and short answers.

These high-quality challenges and datasets have greatly boosted the research progress in machine comprehension, resulting in a wide range of model architectures. Seo et al. (2016) enable a bidirectional attention flow between the question and the context (passage) to obtain question-aware context representations for prediction. Gardner and Clark (2018) focus on machine comprehension with multiple paragraphs with a shared normalization training objective that teaches the model to

predict globally correct answer spans. Hu et al. (2018) enhance the traditional training approach with reinforcement learning and further propose a reattention mechanism that refines the current attention with the past attention. Wang et al. (2017) employ a self-matching mechanism (matching the passage against itself) to refine the passage representation and to encode the information from the whole passage. Huang et al. (2018a) propose a fully-aware attention mechanism that leverages the "history of word" information, which is the information from the lowest word-level embedding up to the highest semantic-level representations. Yu et al. (2018) leverage convolution to model local interactions and self-attention to model global interactions and gain considerable improvement in model efficiency compared to recurrent neural models. In addition to extracting or synthesizing an answer from a passage, another body of work studied multiple-choice machine comprehension (Liu et al., 2020; Chen et al., 2019b), which selects an answer from a candidate set given a question and a passage. This resembles the reading comprehension tasks in school exams, which is less relevant to our information-seeking goal.

The recently proposed BERT (Devlin et al., 2019) model pretrains language representations with bidirectional transformers. It achieves exceptional results on extractive machine comprehension by fine-tuning the task specific layers (a token classification head) on top of the pretrained language model. BERT and other pretrained language models (Lan et al., 2019; Clark et al., 2020; Yang et al., 2019c; Liu et al., 2019b) have become part of the common recipe of many recent machine comprehension models. In Chapters 4 and 5, we also use a BERT-based MC model and study history modeling approaches that can be seamlessly integrated into BERT for conversational question answering.

2.2.3 Open Domain Question Answering

In contrast to the conventional MC tasks that offer a pre-selected passage for answer extraction, open domain QA tasks provide the model with access to a large corpus (Dhingra et al., 2017; Voorhees and Tice, 1999) or at least a set of candidate documents for each question (Nguyen et al., 2016; Joshi et al., 2017; Dunn et al., 2017; Dhingra et al., 2017; Cohen et al., 2018). Some previous work (Lee et al., 2019a; Htut et al., 2018; Kratzwald and Feuerriegel, 2019; Wang et al., 2018) 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. Another body of work, starting from DrQA (Chen et al., 2017), presents end-to-end open-domain QA systems following a retrieve-and-read framework. DrQA (Chen et al., 2017) uses TF-IDF to retrieve from a Wikipedia document collection and then reads the top passages with a multi-layer recurrent neural network machine comprehension model. Similarly, BERTserini (Yang et al., 2019b) uses BM25 as the retriever and a BERT model as the reader.

More recent work has been starting to conduct the retrieval phase with a dense retriever, which is typically a learnable dual-encoder architecture that encodes the question and the passage into low-dimensional dense vectors. Retrieval is then conducted with Maximum Inner Product Search (MIPS) or Approximate Nearest Neighbor (ANN) search. Such retrievers often feature pretrained language models as the encoders. Specifically, ORQA (Lee et al., 2019b) leverages a dual-BERT retriever pretrained with the inverse cloze task (finding the context for a given sentence) and a BERT reader. REALM (Guu et al., 2020) augments ORQA by adding a novel language model pretraining step for the retriever, enabling back-propagation into the MIPS index. ANCE (Xiong et al., 2020) also enhances the retriever training process with ANN retrieved negative passages, which are considered more informative than the in-batch negatives used in ORQA and REALM. Both ANCE and REALM refresh the index asynchronously during training so that the training process is more

effective. Similar to ANCE, DPR (Karpukhin et al., 2020) digs into the negative sampling strategies during training by studying different combinations of in-batch negatives and retrieved negatives. Different from aforementioned research, Das et al. (2019) design the dense retriever and the reader to interact with each other iteratively via query reformulation. ReQA (Ahmad et al., 2019a) also uses a similar dense retriever to retrieve sentence-level answers directly. These learnable dense retrievers have been shown to be highly effective and scalable in open-domain QA. 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 in Chapter 5.

2.3 Conversational Question Answering

Similar to the answer selection and MC tasks in single-turn QAs, existing ConvQA research can be generally classified into two categories, response ranking, and conversational machine comprehension. These tasks are also the testbeds to study history modeling approaches for conversational QA. Our approach in Chapters 4 and 5 belongs to conversational MC.

2.3.1 Response Ranking

The Ubuntu Dialog Corpus (UDC) (Lowe et al., 2015) is one of the most prevalent benchmarks for response ranking. It consists of multi-turn and unstructured dialog data extracted from Ubuntu chat logs. Other popular datasets include the Douban Conversation Corpus (Wu et al., 2016) that contains large-scale open-domain conversations and our MSDialog-ResponseRank (Yang et al., 2018) that features technical support dialogs. A rich body of work studies response ranking with these benchmark datasets. Wu et al. (2016) propose a sequential matching network that first matches a response with each context utterance sequentially and then aggregates these matching signals with a recurrent neural network. Yang et al. (2018) incorporate external

knowledge to the response ranking process with pseudo-relevance feedback and QA correspondence knowledge distillation. Yang et al. (2019a) combine the merits of retrieval-based methods and generation-based methods for a hybrid approach for response ranking. Yang et al. (2020a) introduce an intent-aware neural response ranking model to capture the importance of utterances by leveraging user intent information.

2.3.2 Conversational Machine Comprehension

CoQA (Reddy et al., 2019)[1] and QuAC (Choi et al., 2019)[2] are two large-scale MC-style ConvQA datasets. The ConvQA task in these datasets is very similar to the MC task in SQuAD. A major difference is that the questions in ConvQA are organized in conversations. Although both datasets feature ConvQA in context, they come with different properties. Information-seekers in QuAC have access to the title of the passage only, simulating an information need. QuAC also comes with dialog acts, which is an essential component in this interactive information seeking process. The dialog acts provide an opportunity to study the multi-task learning of answer span prediction and dialog act (user intent) prediction. On the other hand, CoQA offers freeform human-generated answers, making it possible to study the training challenges of weakly-supervised open-retrieval ConvQA. We take advantage of the different unique properties offered by these datasets in our study.

In terms of modeling, Zhu et al. (2018) leverage both self-attention and inter-attention between the question and the passage and enhance this model with contextualized embeddings generated from BERT. Choi et al. (2019) augment the bidirectional attention flow mechanism (Seo et al., 2016) with self attention and contextualized embeddings. They also incorporate conversation history by encoding the dialog turn number within the question embedding and marking the history answers

[1] https://stanfordnlp.github.io/coqa/

[2] http://quac.ai/

in the passage with concatenated marker embeddings. Reddy et al. (2019) adopt the same reader model in DrQA (Chen et al., 2017) and enhance it by prepending history questions and answers. This model also uses a Pointer-Generator network (See et al., 2017) to transform the predicted answer span to a natural answer. Huang et al. (2019) use recurrent structures to integrate the intermediate representations generated when answering previous questions and thus can grasp the latent semantics of the history. Chen et al. (2019a) propose to construct a context graph at each conversation turn that considers the question and the conversation history, followed by a recurrent graph neural network to model the temporal dependencies of these context graphs.

In Chapter 4, we propose a "history selection - history modeling" framework to handle conversation history in ConvQA. In terms of history selection, most existing works mentioned above (Choi et al., 2019; Reddy et al., 2019; Zhu et al., 2018; Huang et al., 2019) adopt a simple heuristic of selecting immediate previous turns. This heuristic, however, might not work well for complicated dialog behaviors. None of this research focuses on *learning* to select or re-weight conversation history turns. To address this issue, we propose a history attention mechanism, which is a learned strategy to attend to history turns with different weights according to how helpful they are on answering the current question. In terms of history modeling, most existing methods at the time typically prepend history turns to the current question (Reddy et al., 2019; Zhu et al., 2018) or use a recurrent structure to model the representations of history turns (Huang et al., 2019), which could have a lower training efficiency. In this dissertation, we propose a history answer embedding method to learn two unique embeddings to denote whether a passage token is in history answers. This method is tailored for BERT-like pretrained language models. We further enhance this method by incorporating the position information into history answer embeddings.

2.4 Session-based Retrieval

The Session Track in TREC focuses on studying information retrieval over user sessions rather than one-time queries (Carterette et al., 2016). Due to the lack of large-scale session datasets, most approaches are mainly limited to non-parametric or feature-based models. Typical techniques include query expansion (Hagen et al., 2013; Cui and Cheng, 2014; Technology et al., 2013) and learning to rank (Cui and Cheng, 2014; Technology et al., 2013). Query expansion methods identify useful terms from the session history, such as those in previous queries, to expand the current query. Learning to rank methods consider the search history by extracting history-specific features, such as the similarity of the candidate document and the session virtual document. These techniques can also be used together. Shen et al. (2005) propose context-sensitive statistical language models to combine the queries and summaries of the clicked documents in the search session. White et al. (2010) study different search behaviors and develop effective weighting schemes to combine the query with these context search behaviors. Xiang et al. (2010) propose context-aware ranking principles and develop context features based on these principles for a learning to rank model. Bennett et al. (2012) investigate how short-term session behavior and long-term historic behavior interact and propose models to learn to combine these features for enhanced performance. In addition to exploiting the current session only, White et al. (2013) work from a different angle by leveraging other similar tasks from historic search logs. Recently, IR researchers have begun to revisit the issue of session-based retrieval with deep models. Ahmad et al. (2018, 2019b) employ hierarchical recurrent structures to model queries and click-through information across history turns. They adopt a multi-task learning setting to optimize for both document ranking and query suggestion. Our work in Chapter 6 focuses on the contextual ranking task with BERT, a pretrained language model, as a further step for conversational search.

2.5 Other Directions in Conversational Search

The area of conversational search is broad and covers many directions beyond the ones mentioned above. We briefly describe more studies in this area to paint a more complete picture of research on conversational search.

The concept of conversational search can be traced back to early research on interactive information retrieval. Oddy (1977) develops the THOMAS system that allows users to conduct searches through man-machine dialogs. Croft and Thompson (1987) introduce a system that is driven by user interactions, such as stating the goals and evaluating system output, at several given stages of a search session. Belkin et al. (1995) explore and demonstrate the justifiability of using information interaction dialogs to design the interaction mechanisms in IR systems. Marchionini (2006) and White and Roth (2009) address the importance of exploratory search, where the behavior of search is beyond a simple look up and more like learning and investigating. In this setting, the interpretation of user intent would rely heavily on the interactions between human and computers.

Recent years have witnessed the revival of conversational search. Researchers have been actively working on the user-oriented aspect for conversational information seeking. Radlinski and Craswell (2017) describe a conceptual framework for conversational IR and the major research issues that must be addressed. Thomas et al. (2017) introduce the MISC dataset that consists of audio and video recordings of information-seeking conversations between human users and intermediaries. Trippas et al. (2017) observe how people conduct mixed initiative conversational search with different cognitive complexity levels in an acoustic setting. They also analyze the initial turns for patterns to classify with a qualitative analysis approach. Chuklin et al. (2019) apply four different prosodic modifications to the voice responses output by a text-to-speech system so that the responses are easier to comprehend by users. Trippas et al. (2018) carry out observational studies of how people conduct search tasks

in an audio-only setting to inform the design of spoken conversational search systems. Qu et al. (2019b) study three answer presentation and interaction approaches in a non-factoid question answering setting. Trippas et al. (2020) create an annotation schema for spoken conversational search data to investigate the interactivity in these conversations.

Another rich body of work targets on the modeling aspect of a variety of conversation tasks. Shah and Pomerantz (2010) consider community QA as an information-seeking process and build models to predict the answer quality. Yang et al. (2017) study neural matching models based on LSTM and CNN for question retrieval and next question prediction. Zhang et al. (2018) explore a "System Ask, User Respond" paradigm for a unified conversational search and recommendation framework. It features a system that can proactively ask aspect-specific questions to understand the user needs. Bi et al. (2019) develop a conversational product search system that is driven by users' positive and negative feedback for the presented items. Aliannejadi et al. (2019) use a question retrieval model and a question selection model to ask clarifying questions proactively and show it boots the retrieval performance dramatically. Zamani et al. (2020) introduce supervised and reinforcement learning models to generate clarifying question, as well as methods to generate candidate answers of these question for users to specify. Hashemi et al. (2020) learn better representations from transformers for conversational search by leveraging external sources, such as top retrieved documents, to guide the training process. The studies presented in our dissertation join the diverse research in this field towards building functional conversational IR systems.

CHAPTER 3

HISTORY MODELING FOR USER INTENT PREDICTION

3.1 Introduction

To build functional and natural conversational assistants that can reply to more complicated tasks, we need to understand how users interact in these information-seeking environments. Thus, it is necessary to analyze and predict user interactions and utterance intent. At the CAIR[1] workshop at SIGIR'17, researchers indicated that there is a lack of conversational datasets to conduct studies. Also, the Learn-IR workshop[2] at WSDM'18 highlighted the significant research need for user intent analysis and prediction in an interactive information-seeking process. Therefore, in this chapter, we address these research demands by a two-part effort. The first part is to create the *MSDialog*[3] dataset with information-seeking conversations and analyze user intent patterns. The second part is to study user intent prediction with MSDialog using both feature-based methods and neural methods.

For effective analysis of user intent in an information-seeking process, the data should be multi-turn information-seeking dialogs. Also, conversational systems should be designed to support natural dialogs. Thus, the data should come from conversation interactions between real humans. As shown in Table 3.1, we found that most existing dialog datasets are not appropriate for user intent analysis. The most similar data to ours is the Ubuntu Dialog Corpus (UDC), which also contains multi-turn

[1] https://sites.google.com/view/cair-ws/

[2] https://task-ir.github.io/wsdm2018-learnIR-workshop/

[3] https://ciir.cs.umass.edu/downloads/msdialog

Table 3.1: Comparison of dialog datasets.

Dataset	Multi-turn	Human-human	Information-seeking	User intent label
DSTC 1-3 (Henderson et al., 2014)	✓			
DSTC 4-5 (Kim et al., 2017)	✓	✓		
Switchboard (Godfrey and Holliman, 1997)	✓	✓		
Twitter Corpus (Ritter et al., 2010)	✓	✓		
DSTC 6 (2nd Track) (Hori and Hori, 2017)	✓	✓	✓	
Ubuntu Dialog Corpus (Lowe et al., 2015)	✓	✓	✓	
MSDialog (ours)	✓	✓	✓	✓

QA conversations in the technical support domain. However, the user intent in this dataset is unlabeled. In addition, UDC dialogs are in the IRC (Internet Relay Chat) style. This informal language style contains a significant amount of typos, internet language, and abbreviations. Another dataset, the DSTC 6 Conversation Modeling track data contains knowledge grounded dialogs from Twitter. However, this dataset contains scenarios where users do not request information explicitly, which do not fit the information-seeking narrative. Thus, these datasets are not appropriate for user intent analysis.

For open-domain chatting, it is common practice to train chatbots with social media data such as Twitter (Ritter et al., 2011). Similarly, real human-human multi-turn QA dialogs are the appropriate data for characterizing user intent in information-seeking conversations. In technical support online forums, a thread is typically initiated by a user-generated question and answered by experienced users (agents). The users may also exchange clarifications with the agents or give feedback based on answer quality. Thus the flow of a technical support thread resembles the information-seeking process if we consider threads as dialogs and posts as turns/utterances in the dialogs. We create MSDialog by crawling multi-turn QA threads from the Mi-

crosoft Community[4] and annotating them with 12 fine-grained user intent types on an utterance level based on crowdsourcing on Amazon Mechanical Turk (MTurk).[5]

With this new dataset, we analyze the user intent distribution, co-occurrence patterns and flow patterns of large-scale QA dialogs. We gain insights on human intent dynamics during information-seeking conversations. One of the most interesting findings is the high co-occurrence of negative feedback and further details, which typically occurs after a potential answer is given. This co-occurrence pattern provides feedback about the retrieved answer and critical information about how to improve the previous answer. In addition, negative feedback often leads to another answer response, indicating that the co-occurrence and flow patterns associated with negative feedback can be the key to iterative answer finding.

We then conduct experiments on user intent prediction with the MSDialog data. In addition, we also annotate a small portion of Ubuntu Dialog Corpus (Lowe et al., 2015) with the same user intent types as in MSDialog to further validate our findings.

The purposes of user intent prediction are threefold. First, it is necessary for conversational assistants to accurately identify user intent in information-seeking conversations. Only in this way can they process the information accordingly and use it to provide answers and adjust previous answers. Similar to customer service over phones, routing user questions to different sub-modules in a conversational retrieval system is only possible if the user intent is correctly identified. Second, the conversational assistants need to learn and imitate the behavior of human agents. By identifying user intent in information-seeking conversations, we expect the conversational assistants to learn the use of different intent and when to issue requests for more information or details spontaneously. Finally, user intent prediction models can

[4] `https://answers.microsoft.com`

[5] `https://www.mturk.com/`

be used to automatically annotate more dialog utterances for data analysis and other tasks such as conversational answer finding.

Previous work typically focused on dialog act classification for open-domain conversations (Stolcke et al., 2000; Liu et al., 2018; Khanpour et al., 2016). In human-computer chitchat, the goal of the conversational assistants is to generate responses that are as realistic as possible with the primary purpose of entertaining. In contrast to chatting, users initiate information-seeking conversations for specific information needs. Human behaviors in chatting and information-seeking conversations can be very different due to the fundamentally distinct purposes. In addition, the Dialog State Tracking Challenges (DSTC)[6] focus on goal-oriented conversations. These tasks are typically tackled with slot filling (Zhang and Wang, 2016; Yan et al., 2017). In information-seeking conversations, slot filling is not suitable because of the diversity of the information needs. User intent analysis and prediction are needed for an information-seeking setting.

We conduct experiments using two different approaches to predict user intent in information-seeking conversations. Firstly, we extract rich features to capture the content, structural, and sentiment characteristics of utterances and learn models with traditional machine learning (ML) methods. Secondly, we use the implicit representation learning in neural architectures to predict user intent without feature engineering. We then build upon the neural models to incorporate context utterances, the utterances preceding and following the current utterance, for enhanced performance. Compared with traditional ML methods, it is easier and more natural for neural models to incorporate context information since they can consume the raw context utterances as input and require no context-specific feature engineering. We present more details in Section 3.6.1.2.

[6] https://www.microsoft.com/en-us/research/event/dialog-state-tracking-challenge/

Our contributions in this chapter can be summarized as follows. (1) We create a large-scale annotated dataset for multi-turn information-seeking conversations, which is the first of its kind to the best of our knowledge. We have made our dataset freely available to encourage relevant studies. (2) We perform in-depth data analysis and characterization of multi-turn human QA conversations. We analyze the user intent distribution, co-occurrence and flow patterns. Our characterizations also hold in similar data (UDC). Our findings could be useful for designing conversational search systems. (3) We extract rich features, including feature groups related to content, structures, and sentiment, to predict user intent in information-seeking conversations. We perform an in-depth feature importance analysis on both group and individual level to identify the key factors in this task. (4) We design several variations of neural classifiers to predict user intent without explicit feature engineering. We show that neural models can achieve comparable performance compared to feature engineering based methods. Moreover, neural models achieve statistically significant improvements over traditional methods after incorporating context information. (5) Our experiments show that the trained model achieves relatively good generalization performance on another open benchmark information-seeking conversation dataset (UDC). The code of the implemented user intent prediction models has been released to the research community.[7]

3.2 The MSDialog Data

Our data collection contains two sets: the complete set and a labeled subset. Both are publicly available. The complete set could be useful for unsupervised/semi-supervised model training. The data used in user intent analysis and prediction is the

[7] `https://github.com/prdwb/UserIntentPrediction`

labeled subset. In this section, we describe the three stages of generating MSDialog, which are data collection, taxonomy definition, and user intent annotation.

3.2.1 Data Collection

We crawled over 35,000 dialogs from Microsoft Community, a forum that provides technical support for Microsoft products. This well-moderated forum contains user-generated questions with high-quality answers provided by Microsoft staff and other experienced users including Microsoft Most Valuable Professionals.

To ensure the quality and consistency of the dataset, we selected about 2,400 dialogs that meet the following criteria for annotation: (1) With 3 to 10 turns. (2) With 2 to 4 participants. (3) With at least one correct answer selected by the community. (4) Falls into one of the categories of Windows, Office, Bing, and Skype, which are the major categories of Microsoft products.

We observe that dialogs with a large number of turns or participants can contain too much noise, while dialogs with limited turns and participants are relatively clean. By choosing dialogs with at least one answer, we can use this dataset for other tasks such as answer retrieval. Also, by limiting the categories to several major ones, we can ensure language consistency across different dialogs, which is better for training neural models.

3.2.2 Taxonomy for User Intent in Conversations

We classify user intent in dialogs into 12 classes shown in Table 3.2. Seven of the classes (*OQ*, *RQ*, *CQ*, *FD*, *PA*, *PF*, *NF*) were first introduced in FIRE'10[8]. Bhatia et al. (2012) added the eighth class of *Junk* as they observed a significant amount of posts with no useful information in their data (200 dialogs labeled with eight classes).

[8] https://www.isical.ac.in/~fire/2010/task-guideline.html

Table 3.2: User intent taxonomy and distribution in MSDialog.

Code	Label	Description	%
OQ	Original Question	The first question from the user to initiate the dialog.	13
RQ	Repeat Question	Other users repeat a previous question.	3
CQ	Clarifying Question	User or agent asks for clarifications.	4
FD	Further Details	User or agent provides more details.	14
FQ	Follow Up Question	User asks for follow up questions about relevant issues.	5
IR	Information Request	Agent asks for information from users.	6
PA	Potential Answer	A potential answer or solution provided by agents.	22
PF	Positive Feedback	User provides positive feedback for working solutions.	6
NF	Negative Feedback	User provides negative feedback for useless solutions.	4
GG	Greetings/Gratitude	Greetings or expressing gratitude.	22
JK	Junk	No useful information in the utterance.	1
O	Others	Utterances cannot be categorized using other classes.	1

We add four more classes to Bhatia et al. (2012)'s taxonomy: *Information Request*, *Follow Up Question*, *Greetings/Gratitude*, and *Others*. We observed that agents' inquiries about user's version of software or model of computer are common in this technical support data and does not necessarily overlap with *Clarifying Question*. *Follow Up Question* is another utterance class in MSDialog as users sometimes expect agents to walk them step-by-step through the technical problem. *Greetings/Gratitude* is quite common in the data. Finally, the *Others* class is for utterances that cannot be classified with other classes. Note, each utterance can be assigned multiple labels because an utterance can cover multiple intent (e.g., *GG+FQ*).

3.2.3 User Intent Annotation with MTurk

3.2.3.1 Procedure

We employed crowdsourcing workers through MTurk to label user intent of each utterance using a set of 12 labels that is described in Section 3.2.2. The workers are required to have a HIT (Human Intelligence Task) approval rate of 97% or higher, a minimum of 1,000 approved HITs, and be located in the US, Canada, Australia or Great Britain. The workers are paid $0.3/dialog.

Table 3.3: Dialog properties of MSDialog

Items	Min	Max	Mean	Median
# Turns Per Dialog	3	10	4.56	4
# Participants Per Dialog	2	4	2.79	3
Dialog Length (Words)	27	1,467	296.90	241
Utterance Length (Words)	1	939	65.16	47

In this annotation task, the workers are provided with a complete dialog. They are instructed to go through a table of labels with descriptions and examples before they proceed. For each utterance, the workers are tasked to choose all applicable labels that represent the user intent of the utterance and leave a comment if they choose the *Others* label.

3.2.3.2 Quality Assurance

To ensure the annotation quality, we employed two workers on each dialog. We calculated the inter-rater agreement using Fuzzy Kappa (Kirilenko and Stepchenkova, 2016) for this one-to-many classification task. We applied the threshold of 0.18 to filter the dialogs with too small Kappa scores, which reduced the number of dialogs by 9%.

3.3 User Intent Analysis and Characterization

3.3.1 Data Statistics

The annotated dataset contains 2,199 multi-turn dialogs with 10,020 utterances. Table 3.3 summarizes the properties of MSDialog. Each utterance has 1.83 labels on average. More statistics of the dataset are presented in Table 3.4.

3.3.2 User Intent Distribution

Figure 3.1 shows the user intent distribution. Labels without a percentage are under 10%. *Greetings/Gratitude* and *Potential Answer* are the most frequent labels.

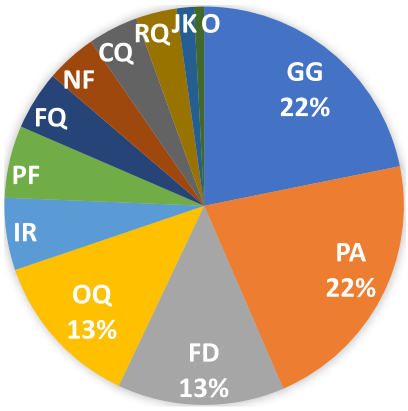


Figure 3.1: User intent distribution

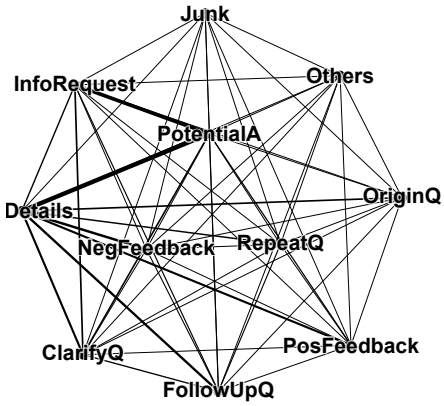


Figure 3.2: User intent co-occurrence

This suggests that good manners and answers are at the center of human QA conversations. *Repeat Question* is the most infrequent label except for *Junk* and *Others*, which is because the number of participants is limited to four.

3.3.3 User Intent Co-occurrence

Label co-occurrence in the same utterance can be useful for understanding user intent. Preliminary results indicate that the most frequent co-occurrence is between *Greetings/Gratitude* and another label, suggesting good manners of forum users. Nevertheless, we removed *GG* for the analysis later to emphasize more on crucial user intent of information-seeking interactions.

We present the user intent co-occurrence graph with undirected edges weighted by co-occurrence count in Figure 3.2. We observe that *Potential Answer* often co-occurs with *Further Details* or *Information Request*. This indicates that agents tend to enrich possible solutions with details, or send *Information Request*s in case the solutions do not work. Also, users tend to give *Negative Feedback* with *Further Details* to explain how the suggested answer is not working. In addition, *Further Details* is observed to co-occur with *Follow Up Question* or *Clarifying Question*, suggesting that when people raise a relevant question, they tend to add details to them.

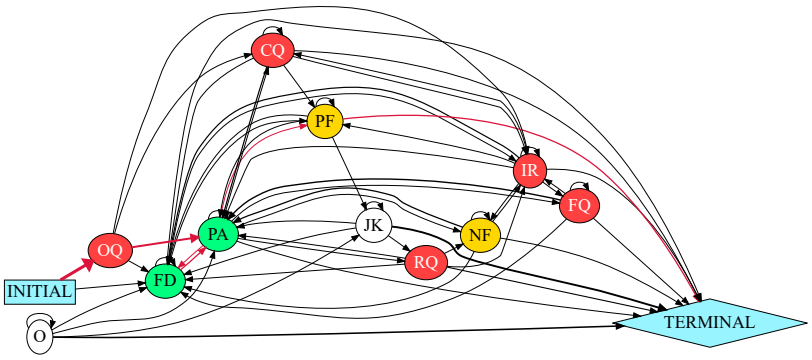


Figure 3.3: Flow pattern with a Markov model. Node colors: red (questions), green (answer related), yellow (feedback). Edges are directed and weighted by transition probability.

3.3.4 User Intent Flow Pattern

We use a Markov Model to analyze the flow patterns in the dialogs as shown in Figure 3.3. Because of the complexity and diversity of human conversations, many utterances are labeled with multiple user intent. We preprocess the traces (complete user intent flow in a dialog) with multiple labels by only using one label each time. For example, if we have a trace of "$OQ \rightarrow PA+FD \rightarrow PF$", we transfer it into two separate traces. The first one is "$OQ \rightarrow PA \rightarrow PF$", and the second one is "$OQ \rightarrow FD \rightarrow PF$". This preprocessing step can lead to a more concise model compared with using the original multi-labels as nodes. However, it does magnify some user intent non-proportionally. We alleviate the issue by only using dialogs that generate no more than 100 traces. This only filtered 30 dialogs.

In addition, we remove *Greetings/Gratitude* because of the same reason described in Section 3.3.3. Instead of simply hiding the *GG* node from the final graph, we remove the occurrences of *Greetings/Gratitude* if the utterance has multiple labels or change *GG* to *JK* if the utterance only has one label.

The flow pattern with a Markov model is presented in Figure 3.3. As highlighted in the graph, a typical user intent transition path of MSDialog is "$INITIAL \rightarrow OQ \rightarrow PA \rightarrow FD \rightarrow PA \rightarrow PF \rightarrow TERMINAL$". This represents the frequent user

intent transition pattern in an information seeking process. We can make some observations from the graph : (1) In most cases, dialogs begin with an *Original Question*, sometimes accompanied by *Further Details*. (2) *Original Question* tends to lead to *Potential Answer* and *Information Request*. (3) *Information Request* and *Clarifying Question* tend to lead to *Further Details*. (4) *Positive Feedback* tends to terminate the dialog while *Negative Feedback* tends to lead to *Potential Answer* or *Further Details*. (5) Dialogs tend to end after *Others* or *Junk*.

Besides the Markov transition graph, we use a different perspective to inspect the flow pattern by focusing on the user intent transition between turns in each dialog. We find that a quite significant flow path across turns is "$INITIAL \rightarrow OQ \rightarrow (PA \rightarrow FD) \times 3 \rightarrow PA \rightarrow PF \rightarrow TERMINAL$". The "$PA \leftrightarrow FD$" circle pattern is typically caused by the "$PA+IR$", "$PA+CQ$", "$NF+FD$" co-occurrences described in Section 3.3.3 and the "$IR \rightarrow FD$", "$CQ \rightarrow FD$", "$NF \rightarrow PA$" sequential relationship suggested in Figure 3.3.

3.3.5 Comparison with Ubuntu Dialog Corpus

Although UDC is less suitable for user intent analysis due to the informal language style, we investigate the characterizations of UDC and compare them to MSDialog since they are both in the technical support domain. We sampled 200 UDC dialogs and annotated user intent with MTurk using the same method with MSDialog. The informal language style of UDC may impact the annotation quality.

3.3.5.1 Statistics

For this section, we present the statistics for UDC (complete set) and MSDialog. As shown in Table 3.4, UDC dialogs have shorter utterances because of the informal language style.

Table 3.4: Statistics of UDC and MSDialog

Items	UDC	MSDialog (complete)	MSDialog (labeled)
# Dialogs	930,000	35,000	2,199
# Utterances	7,100,000	300,000	10,020
# Words (in total)	100,000,000	24,000,000	653,000
Avg. # Participants	2	3.18	2.79
Avg. # Turns Per Dialog	7.71	8.94	4.56
Avg. # Words Per Utterance	10.34	75.91	65.16

3.3.5.2 Data Characterization

Potential Answer and *Further Details* are the most significant user intent in UDC, which is consistent with MSDialog. Interestingly, the most common user intent in MSDialog, *Greetings/Gratitude*, is quite rare in UDC. In addition, we observe the exact same top 5 label co-occurrences in UDC as described in Section 3.3.3. Note that they are not necessarily in the same order. Finally, we found that the flow patterns observed in MSDialog also hold in UDC, except for the tendency from *Positive Feedback* to *TERMINAL*. This can be explained by the scarcity of *Positive Feedback* in UDC. Although the UDC dialogs with informal language style are drastically different from the formal written style of MSDialog, the resemblance in user intent characterizations indicates that human QA conversations, regardless of the communication medium, follow similar patterns.

3.3.6 Discussion

In this section, we discuss the limitations of our findings. The patterns we discovered are closely related to several design choices, including using dialogs from a well moderated forum in a specific domain. These choices were made to keep the setting as clean as possible as the research community is at an initial stage of this study. Although MSDialog does not cover every aspect of the highly diverse information-seeking conversations, it should be a first step to analyze and predict user intent in an information-seeking setting.

3.4 The User Intent Prediction Task

3.4.1 Task Definition

The research problem of user intent prediction in information-seeking conversations is defined as follows. The input of the system is an information-seeking dialog dataset $\mathcal{D} = \{(\mathcal{U}_i, \mathcal{Y}_i)\}_{i=1}^{N}$ and a set of user intent labels $\mathcal{L} = \{l_1, l_2, \ldots, l_c\}$. A dialog $\mathcal{U}_i = \{u_i^1, u_i^2, \ldots, u_i^k\}$ contains multiple turns of utterances. u_i^k is the utterance at the k-th turn of the i-th dialog. $\mathcal{Y}_i$ consists of annotated user intent labels $\{\mathbf{y}_i^1, \mathbf{y}_i^2, \ldots, \mathbf{y}_i^k\}$, where $\mathbf{y}_i^k = \{y_i^{k(1)}, y_i^{k(2)}, \ldots, y_i^{k(c)}\}$. Here $y_i^{k(m)}, \ldots, y_i^{k(n)} = 1$ denotes that the utterance u_i^k in dialog $\mathcal{U}_i$ is labeled with user intent $\{l_m, \ldots, l_n\}$. Given an utterance u_i^k and other utterances in dialog $\mathcal{U}_i$, the goal is to predict the user intent $\mathcal{Y}_i$ of this utterance. The challenge of this task lies in the complexity and diversity of human information-seeking conversations, where one utterance often expresses multiple user intent (Trippas et al., 2018).

3.4.2 Dataset

We use the labeled subset of the MSDialog data introduced in Section 3.2. In order to test the generalization performance of our findings, we use a small portion of UDC that is annotated with the same user intent types as in Section 3.2.2. This part of experiment is presented in Section 3.6.3.

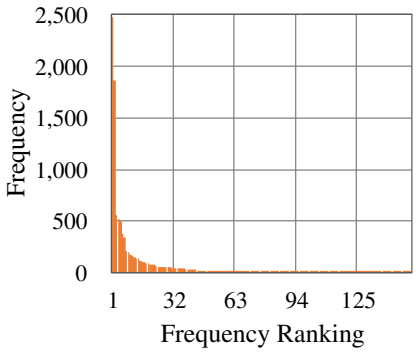
3.4.3 Data Preprocessing

The purpose of this classification task is to identify and predict user intent so that conversational assistants can process the information accordingly to satisfy the users' information needs. However, utterances which were labeled *Greetings/Gratitude*, *Junk*, and *Others* do not contribute to the purpose of providing information about QA related user intent. Therefore, we remove occurrences of these labels. Note that we only remove these labels if there are more than one label of the given utterance. For example, if the annotation for the given utterance is *GG+OQ*, we transform the

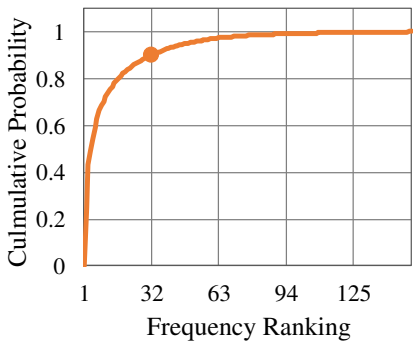
annotation into *OQ*. If the annotation is just *GG*, no transformation is needed. This reduces the number of unique label combinations from 316 to 152.

In addition, some label combinations of user intent labels are quite rare in the data. As indicated in Figure 3.4a, the top frequent label combinations have hundreds of occurrences in the data (e.g. *PA*, *OQ*, *PF*, *FD+PA*, *FD*), while the least frequent labels only have exactly one occurrence (e.g. *CQ+FD+IR+RQ*, *CQ+FD+FQ+PF*). These rare label combinations are very likely due to minor annotation errors or noise with MTurk. Annotation quality assurance was performed based on the dialog-level inter-rater agreement to keep the complete dialog intact and thus may result in minor noise on an utterance level. We also plot the cumulative distribution of label combinations for better illustration in Figure 3.4b. The most frequent 32 label combinations constitute 90% of total label combination occurrences as marked in the figure. All 12 user intent labels are individually present in these 32 most frequent combinations except for *Others*. For the rest of the label combinations, we randomly sample one of the labels from each combination as the user intent label for the given utterance. For example, if the annotation for the given utterance is *CQ+FD+IR+RQ*, we transform it into a single label by randomly sampling one of the four labels, such as *CQ*. Therefore, the total number of label combinations in the data was reduced to 33 (including *Others*). We adopt this setting since these rare label combinations are very likely due to minor annotation errors. In addition, it would be very difficult to learn a prediction model for these label combinations with few instances.

For UDC, we observe a similar label combination distribution. So we preprocess the data in the same way. We have 34 label combinations for the UDC with 27 of them overlapping with MSDialog.



(a) Label combination occurrence from the most frequent to the least frequent.



(b) Cumulative occurrence probability added from the most frequent to the least frequent. The marked point is $(32, 0.9)$.

Figure 3.4: Intent label combination distribution

3.5 User Intent Prediction with Feature-based Methods

In this section, we extract several features following previous work (Bhatia et al., 2012; Ding et al., 2008) and adopt different ML methods to build baseline models. In addition to reporting baseline performance, we also perform feature importance analysis to identify key factors in user intent prediction.

3.5.1 Features

We extract three groups of features to detect user intent in information-seeking conversations, including content, structural, and sentiment features. An overview of the features is provided in Table 3.5. Although MSDialog is derived from forum data, it is considered as a dialog dataset. Thus, we refrain from developing features that can only be extracted from the metadata of the forum, such as user authority level or answer votes, so that our method can be applied to dialog systems. The features are designed to capture the content and sentiment characteristics of the utterances as well as the structural information of the dialogs.

Content features. We build a TF-IDF representation of utterances and compute the cosine similarity of the given utterance with the dialog initial utterance (which

Table 3.5: Features extracted for user intent prediction in information-seeking conversations. "Str", "Con", "Sen" refer to "Structural", "Content", "Sentiment" respectively. "R", "B", "O", "N" refer to "Real", "Binary", "One-hot", "Nuemerical" respectively.

Feature Name	Group	Description	Type
Initial Utterance Similarity	Con	CosSim between the utterance and the first utterance	R
Dialog Similarity	Con	CosSim between the utterance and the entire dialog	R
Question Mark	Con	Does the utterance contain a *question mark*	B
Duplicate	Con	Does the utterance contain *same*, *similar*	B
5W1H	Con	Does the utterance contain *what*, *where*, *when*, *why*, *who*, *how*	O
Absolute Position	Str	Absolute position of an utterance in the dialog	N
Normalized Position	Str	Absolute Position divided by # utterances	R
Utterance Length	Str	Total number of words in an utterance after stop words removal	N
Utterance Length Unique	Str	Unique # words in an utterance after stop words removal	N
Utterance Length Stemmed Unique	Str	Unique # words in an utterance after stop words removal and stemming	N
Is Starter	Str	Is the utterance made by the dialog starter	B
Thank	Sen	Does the utterance contain the keyword *thank*	B
Exclamation Mark	Sen	Does the utterance contain an *exclamation mark*	B
Feedback	Sen	Does the utterance contain the keyword *did not*, *does not*	B
Sentiment Scores	Sen	Sentiment scores by VADER (Hutto and Gilbert, 2014)	R
Opinion Lexicon	Sen	Number of positive and negative words from an opinion lexicon	N

typically is the question that initiates the QA dialog), and the entire dialog. These features are meant to capture the relevance level of the given utterance to the dialog in a general way. In addition, the presence of question marks is a strong indicator that the current utterance contains a question. Moreover, we assume that 5W1H keywords (what, where, when, why, who, and how) can suggest the type of the question.

Structural features. The position of an utterance in a dialog can reveal crucial information about user intent. Intuitively, answers tend to be at even number positions in a dialog, while user feedback and follow up questions tend to be at odd number positions. In addition, we include the utterance length with and without duplication removal and stemming. We analyzed the data and found that utterances containing positive feedback are relatively short, while utterances containing questions or answers tend to be long as they typically contain details. Finally, if the given utterance is generated by the information seeker (dialog starter), it is more likely to contain user related questions or user feedback. The structural features not only provide individual characteristics for utterances, but also evaluate the utterances on a dialog level.

Sentiment features. We expect sentiment features to be useful in identifying user feedback and gratitude expressions. In information-seeking conversations, posi-

tive and negative sentiments do not necessarily determine the feedback type. However, we expect them to be correlated to some extent. We also include classic indicators of sentiment, such as the presence of "thank", "does not/did not" and exclamation marks. In addition, we use VADER (Hutto and Gilbert, 2014) to compute the positive/negative/neutral sentiment scores. We also count the number of positive and negative words using an opinion lexicon (Liu et al., 2005).

3.5.2 Methods and Evaluation Metrics

3.5.2.1 Methods

For each utterance, we extract a set of features as described in Section 3.5.1. To apply traditional ML methods with features to this task, we need to transform this multi-label classification problem to multi-class classification. Three transformation strategies are typically used: binary relevance, classifier chains, and label powerset. Binary relevance does not consider the label correlations and label powerset generates new labels for every label combination. So we choose classifier chains as the transformation strategy for traditional ML methods. This strategy performs binary classifications for each label and take predictions for previous labels as extra features. This transformation strategy is the best fit for our task as it considers the label dependency without explicitly generating new labels for every label combination. We adopt classic ML methods, including Naive Bayes classifier, SVM, random forest, and AdaBoost as baseline classifiers. In addition, we use ML-kNN, which supports multi-label classification by nature.

3.5.2.2 Metrics

Due to the nature of the intent prediction task, we adopt metrics suitable for multi-label classification problems as follows.

Accuracy (Acc). It is known as the intersection over the union (IoU) in the multi-label classification settings. Accuracy is defined as the number of correctly

Table 3.6: Statistics of training, validation, and testing sets

Item	Train	Val	Test
# Utterances	8,064	986	970
Min. # Turns Per Dialog	3	3	3
Max. # Turns Per Dialog	10	10	10
Avg. # Turns Per Dialog	4.58	4.48	4.43
Avg. # Words Per Utterance	70.42	67.53	68.64

predicted labels divided by the union of predicted and true labels for every utterance ($\frac{TP}{TP+FP+FN}$, where TP, FP, and FN refer to True Positive, False Positive, and False Negative respectively). For example, if the model predicts "*PA+CQ*" for the given utterance while the ground truth is "*PA+IR*", then the accuracy is $\frac{1}{3}$. The reported performance is the average metric over all utterances.

Precision, recall and F1 score. Precision is defined as the number of correctly predicted labels divided by predicted labels. Recall is defined as the number of correctly predicted labels divided by true labels. F1 is their harmonic mean. These metrics provide an overall performance evaluation for all utterances.

3.5.3 Main Experiments and Results

3.5.3.1 Experimental Setup

We split the labeled subset of MSDialog into training, validation, and test sets. Table 3.6 gives the statistics of the three sets. The models are trained with scikit-multilearn[9] and scikit-learn[10] on the training set. We tune the hyper-parameters on the validation set based on accuracy and report the performance on the test set.

[9] http://scikit.ml/

[10] http://scikit-learn.org/stable/

Table 3.7: Experiment results for baseline classifiers

Methods	Acc	Precision	Recall	F1
ML-kNN	0.4715	0.6322	0.4471	0.5238
NaiveBayes	0.4870	0.5563	0.4988	0.5260
SVM	0.6342	0.7270	0.5847	0.6481
RandForest	0.6268	**0.7657**	0.5903	**0.6667**
AdaBoost	**0.6399**	0.7247	**0.6030**	0.6583

3.5.3.2 Baseline Results

The baseline results are presented in Table 3.7. Two ensemble methods, random forest and AdaBoost achieve the best overall performance of all baseline classifiers. AdaBoost achieves the best accuracy while random forest achieves the best F1 score. Surprisingly, ML-kNN performs relatively poorly despite its nature of an adapted algorithm for multi-label classification.

3.5.4 Additional Feature Importance Analysis

3.5.4.1 Feature Group Analysis

We use one of the best baseline classifiers, random forest, and different combinations of feature groups to analyze the feature importance on a group level. The hyper-parameters are set to the best ones tuned on all features.

For using a single feature group, structural features is the most important feature group as presented in Table 3.8. Structural features and content features are significantly more important than sentiment features. We expect the sentiment features to capture the sentiment in user feedback but they might not be able to effectively discriminate other user intent. Structural features provide better performance than content features. We believe that this can be explained by the fact that handcrafted content features cannot capture the complex user intent dynamics in human information-seeking conversations.

Table 3.8: Experiment results for different feature groups

Group(s)	Acc	Precision	Recall	F1
Content	0.5272	0.6097	0.4821	0.5384
Structural	**0.5809**	**0.6871**	**0.5434**	**0.6068**
Sentiment	0.3306	0.4087	0.3222	0.3603
Con+Str	0.6081	0.7393	0.5640	0.6399
Con+Sen	0.5577	0.6523	0.5179	0.5774
Str+Sent	**0.6110**	**0.7569**	**0.5672**	**0.6485**
All	**0.6268**	**0.7657**	**0.5903**	**0.6667**

For combinations of two feature groups, content+structural features and structural+sentiment features achieve comparable results. However, structural+sentiment features achieve slightly higher results on all metrics. The performance of using two groups of features is higher than using one of these two feature groups individually. Thus, combining structural features with another feature group boosts the performance of using structural features alone. Interestingly, content+sentiment features is unable to outperform the structural features alone. The results of using all features is the highest among all settings, confirming that all feature groups contribute to the performance of user intent prediction.

3.5.4.2 Feature Importance Scores

In the previous section, we evaluated the feature importance on a group level. In this section we focus on individual features to provide a more fine-grained analysis. We use random forest to output individual feature importance scores.[11] As described in Section 3.5.2, we used classifier chains to transform this multi-label classification problem. This method expands the feature space by including previous label predictions as new features for the current label prediction. This makes it not appropriate

[11] https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

to evaluate original features. Thus, we use the Label Powerset method as the data transformation strategy for this section. The relative feature importance scores are presented in Table 3.9. This analysis can identify key factors in user intent prediction.

Table 3.9: Individual feature importance from a random forest classifier with relative importance scores. "Str", "Con", "Sen" refer to "Structural", "Content", "Sentiment" respectively.

Rank	Feature	Group	Impt	Rank	Feature	Group	Impt
1	AbsPos	Str	1.0	13	Lex(Pos)	Sen	0.2814
2	InitSim	Con	0.9745	14	Lex(Neg)	Sen	0.2337
3	NormPos	Str	0.8684	15	Thank	Sen	0.1607
4	Starter	Str	0.8677	16	How	Con	0.08074
5	DlgSim	Con	0.6778	17	Dup	Con	0.06908
6	SenScr(Neu)	Sen	0.6465	18	What	Con	0.06576
7	SenScr(Pos)	Sen	0.5601	19	ExMark	Sen	0.06424
8	Len	Str	0.5335	20	When	Con	0.05989
9	LenUni	Str	0.4381	21	Feedback	Sen	0.02859
10	LenStem	Str	0.4354	22	Where	Con	0.02356
11	SenScr(Neg)	Sen	0.3495	23	Why	Con	0.0232
12	QuestMark	Con	0.3003	24	Who	Con	0.01423

We summarize our observations as follows: (1) Structural features including "Absolute Position", "Normalized Position", "Is Starter" are ranked in the top-5 in terms of feature importance. Moreover, other structural features, such as various forms of utterance length are observed to be relatively informative in general. This confirms the results in Section 3.5.4.1 that the structural feature group is the most important one. (2) "Initial Utterance Similarity" and "Dialog Similarity" are content features that can be highly informative for identifying user intent. Both features are indicators of how closely the utterance connects with the information-seeking process. Other content features, such as "5W1H", however, contribute little to predicting user intent. (3) Some sentiment features are relatively important in identifying user intent, such as positive and neutral sentiment scores. However, some other sentiment features contribute little to the task, such as the existence of exclamation marks and

"thank". (4) We observe that features ranked from the 15$^{\text{th}}$ to the last one in Table 3.9 are all "keyword features". These features are based on a simple rule that whether the given utterance contains pre-defined keywords. For example, the "5W1H" feature looks for "what/where/when/why/who/how" in the given utterance and the "Feedback" feature looks for "did not/does not". The major drawback of manual feature engineering is amplified in this task due to the complexity and diversity of human information-seeking conversations.

3.6 User Intent Prediction with Enhanced Neural Classifiers

We expected the content of an utterance to be a good indicator of user intent types compared to other features. However, as shown in Section 3.5.4, the handcrafted content features are unable to capture the complex characteristics of human information-seeking conversations. Thus, in this section we adopt neural architectures to automatically learn representations of utterances without feature engineering.

3.6.1 Our Approach

3.6.1.1 Base Models

Given the previous success in modeling text sequences using CNN and bidirectional LSTM (BiLSTM) (Graves and Schmidhuber, 2005), we choose these two architectures as our base models. Although utterances are grouped as dialogs, the base models consider utterances independently.

Given an utterance $u_i^k = \{w_1, w_2, \ldots, w_m\}$ (the k-th utterance in the i-th dialog) that contains m tokens, we first transform the sequence of tokens into a sequence of token indices $\boldsymbol{S} = \{s_1, s_2, \ldots, s_m\}$. Then we pad the sequence $\boldsymbol{S}$ to a fixed length n (the max sequence length). Both CNN and BiLSTM start with an embedding layer initiated with pretrained word embeddings. Preliminary experiments indicated that using MSDialog (complete set) to train word embeddings is more effective than using

GloVe (Pennington et al., 2014) in terms of final model performance. The embedding layer maps each token in the utterances to a word embedding vector with a dimension of d.

We focus on the CNN model following previous work (Kim, 2014) here, because it achieves better performance in our experiments. After the embedding layer, filters with the shape (f, d) are applied to a window of f words. f is also referred to as the filter size. Concretely, a convolution operation is denoted as

$$c_i = \sigma(\mathbf{w} \cdot \mathbf{e}_{i:i+f-1} + b) \tag{3.1}$$

Where c_i is the feature generated by the i-th filter with weights $\mathbf{w}$ and bias b. This filter is applied to an embedding matrix, which is the concatenation from the i-th to the $(i + f - 1)$-th embedding vectors. An non-linearity function (ReLU) is also applied. This operation is applied to every possible window of words and generates a feature map $\mathbf{c} = \{c_1, c_2, \ldots, c_{n-f+1}\}$. More filters are applied to extract features of the utterance content. Max pooling are applied to select the most salient feature of a window of f' features by taking the maximum value $\hat{c}_i = max\{\mathbf{c}_{i:i+f'-1}\}$. f' denotes the max pooling kernel size. A dropout layer is applied after the pooling layer for regularization.

After the last convolutional layer, we perform global max pooling by taking the maximum value $\hat{c} = max\{\mathbf{c}\}$ for the feature map $\mathbf{c}$ at this step. This operation reduces the dimension of the tensor to one. This tensor is further transformed to an output tensor of shape $(1, l)$, where l is the number of user intent labels (12 for our task). Sigmoid activation is applied to each value of the output tensor to squash the value to a confidence level between 0 and 1. A threshold is chosen to determine whether the given label present in the final prediction. If the model is not confident of predicting any label, the label of the highest confidence level is the prediction. We tuned the threshold with the validation data.

3.6.1.2 Incorporate Context Information

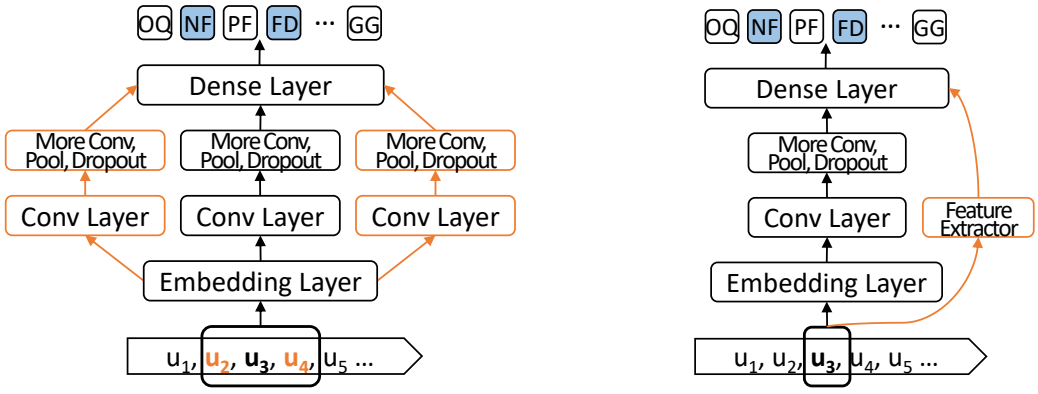
As shown in Section 3.3, user intent follows clear flow patterns in information-seeking conversations. The user intent of a given utterance is closely related to the utterances around it, which compose the *context* for the given utterance. Incorporating context information into traditional ML methods requires additional feature engineering to develop context-specific features (e.g., the term overlap between the previous utterance and the current utterance). In contrast, it is easier and more natural for neural models to incorporate context information since they can consume the raw context utterances as input and require no feature engineering. Therefore, we only focus on context incorporation for neural models in this dissertation. We consider two ways as follows.

Direct Expansion. The most straightforward way to incorporate context information is to expand the given utterance with its context. Concretely, the expanded utterance for u_i^k is $\hat{u}_i^k = u_i^{k-1} \oplus u_i^k \oplus u_i^{k+1}$, where $\oplus$ is the concatenate operator. $\hat{u}_i^k$ is considered as the given utterance in base models.

Context Representation. Given an expanded utterance $\hat{u}_i^k$ as input, the neural architecture first segments it into three original utterances of u_i^{k-1}, u_i^k, and u_i^{k+1}. We apply convolution operations and max pooling to the utterances separately as shown in Figure 3.5a. After global pooling following the last convolutional layer, the three one-dimensional tensors are concatenated for final predictions. This approach extracts features from the given utterance and its context separately. Thus, we are able to learn the importance of the given utterance and its context by tuning context-specific hyper-parameters, such as the number of filters for context utterances.

3.6.1.3 Incorporate Extra Features

We found many useful features for user intent prediction such as structural features in the feature importance analysis in Section 3.5.4. However, nearly half of the



(a) CNN-Context-Rep (b) CNN-Feature

Figure 3.5: Architectures for enhanced neural classifiers. Components marked orange are extra information incorporated into the base CNN model (black). The utterance in bold is the current utterance. Predicted labels are marked blue.

features cannot be exploited by only looking at a single utterance. For example, normalized/absolute utterance position and utterance similarity with the dialog/initial utterance cannot be captured without a holistic view over the entire dialog. Some of the uncaptured features are highly informative. This motivates us to incorporate hand-crafted features into the neural architectures. As shown in Figure 3.5b, all hand-crafted features described in Section 3.5.1 are incorporated into neural architectures at the last dense layer. The feature vector is concatenated with the neural representation of the utterance before making final predictions.

Finally, we combine two base models with various extra components to produce several systems for comparison as follows:

- **CNN**. The base CNN model that consists of three convolutional layers with the same filter size.
- **CNN-Feature**. The CNN model that incorporates extra hand-crafted features at the last layer.

- **CNN-Context**. The CNN model that incorporates context information with direct expansion.
- **CNN-Context-Rep**. The CNN model that incorporates context information with context representation.
- **BiLSTM**. The BiLSTM model that represents the given utterance both in the ordinary order and the reverse order.
- **BiLSTM-Context**. The BiLSTM model that incorporates context information with direct expansion.

3.6.2 Experiments and Evaluation

3.6.2.1 Neural Baselines

In addition to the base model of **BiLSTM** and **CNN**, we further introduce two commonly used neural models for text classification as baselines. For both new baselines, we modify the models to generate multi-label predictions.

CNN-MFS. The CNN model with multiple filter sizes as described by Kim (2014) is a pioneer model to apply neural networks to text classification. This model uses different filter sizes of 3, 4, and 5 to generate feature maps of different window sizes.

Char-CNN. Zhang et al. (2015) introduced a character-level CNN for text classification. There are two variants of the model, a large one and a small one, depending on the numbers of convolutional filters and dense layer units. We report the performance on the small model as it achieves better results in our task.

3.6.2.2 Experimental Setup

We use the same data and metrics as in baseline experiments in Section 3.5. All models are implemented with TensorFlow[12] and Keras.[13] Hyper-parameters are tuned with the validation data in a grid-search style. We found that setting (convolutional filters, dropout rate, dense layer units, max sequence length, convolutional filters for context, and dense layer units for context) to (1024, 0.6, 256, 800, 128, 128) respectively turned out to be the best setting for our best performing model CNN-Conext-Rep. The convolutional filter size and pooling size are set to (3, 3). All models are trained with a NVIDIA Titan X GPU using Adam (Kingma and Ba, 2015). The initial learning rate is 0.001. The parameters of Adam, β_1 and β_2 are 0.9 and 0.999 respectively. The batch size is 128. For the word embedding layer, we trained word embeddings with Gensim[14] with CBOW model using MSDialog (complete set). The dimension of word embedding is 100. Word vectors are set to trainable.

3.6.2.3 Evaluation Results

We select the two strongest feature based classifiers from Section 3.5 as feature based baselines in addition to neural baselines. They are random forest with the best F1 score and AdaBoost with the best accuracy. The performance comparison of models is presented in Table 3.10.

The base CNN model without feature engineering achieves similar results with the strongest feature based baselines. The performance of the base CNN model is better than the base BiLSTM model. CNN-MFS takes advantage of different filter sizes and also achieves comparable results. Char-CNN, however, performs poorly in this task.

---

[12] `https://www.tensorflow.org/`

[13] `https://keras.io/`

[14] `https://radimrehurek.com/gensim/`

Table 3.10: Results comparison. The significance test can only be performed on accuracy. In a multi-label classification setting, accuracy gives a score for each individual sample, while other metrics evaluate the performance over all samples. ‡ means statistically significant difference over the best baseline with $p < 0.01$ measured by the Student's paired t-test.

Method Types	Methods	Accuracy	Precision	Recall	F1
Feature based Baselines	Random Forest	0.6268	**0.7657**	0.5903	**0.6667**
	AdaBoost	**0.6399**	0.7247	**0.6030**	0.6583
Neural Baselines	BiLSTM	0.5515	0.6284	0.5274	0.5735
	CNN	**0.6364**	0.7152	**0.6054**	**0.6558**
	CNN-MFS	0.6342	**0.7308**	0.5919	0.6541
	Char-CNN	0.5419	0.6350	0.4940	0.5557
Neural Classifiers	BiLSTM-Context	0.6006	0.6951	0.5640	0.6227
	CNN-Feature	0.6509	0.7619	0.6110	0.6781
	CNN-Context	0.6555	0.7577	0.6070	0.6740
	CNN-Context-Rep	**0.6885**‡	**0.7883**	**0.6516**	**0.7134**

Char-CNN does not use pretrained word embeddings because it learns features from a character-level which would require much more training data.

BiLSTM performs poorly in this task. Even though BiLSTM-Context has a major improvement over BiLSTM, it has inferior results compared to the base CNN model. Compared to non-factoid question answering or chatting, utterances in information-seeking conversations tend to be longer. BiLSTM(-Context) tries to model a holistic sequence dependency and thus performs poorly on handling these long utterances.

The best result of the feature based baselines is slightly higher than neural baselines. This can be accounted for by the lack of information in neural models. Even though we assume that most of the content and sentiment features can be learned by neural models, the neural models have no access to most of the structural features. Thus, we incorporate all the features to neural models to produce CNN-Feature model. This model gives higher results than baseline classifiers, indicating that incorporating dialog-level information could be beneficial to predicting user intent.

Both CNN-Context and CNN-Context-Rep outperform baseline models and CNN-Feature without explicit feature engineering. These results demonstrate the effectiveness of the implicit feature learning of neural architectures. CNN-Context-Rep performs better than CNN-Context. This indicates that incorporating high-level features of context information learned by neural architectures is better than directly capturing the raw context information. Based on how we compute the metrics (Section 3.5.2.2), accuracy produces a score for each individual sample, while precision/recall/F1 evaluate the performance over all samples. Thus, accuracy is the only metric that is suitable for significance tests. Our best model, CNN-Context-Rep achieves statistically significant improvement over the best baseline with $p < 0.01$ measured by the Student's paired t-test.

3.6.3 Generalization on Ubuntu Dialogs

In this section, we would like to evaluate the generalization performances of different methods on other data in addition to MSDialog. We train different models on MSDialog and test them on the Ubuntu Dialog Corpus (UDC). We select the two best performing feature based classifiers (random forest and AdaBoost) and the best neural model (CNN-Context-Rep) to test the generalization performance. Although the number of annotated Ubuntu dialogs is limited, it is sufficient to demonstrate the predicting performance. We split the annotated UDC data into validation and test sets with an equal size. We train the model on MSDialog data only and tune the hyper-parameters on the UDC validation set. The performance on the UDC test set is presented in Table 3.11.

The generalization results on UDC are not as good as that on MSDialog. Although MSDialog and UDC both consist of multi-turn information-seeking dialogs from the technical support domain, the drastically different language style adds difficulty for model generalization and transferring. In this challenging setting, CNN-Context-

Table 3.11: Testing performance on UDC of different models trained with MSDialog. The significance test can only be performed on accuracy. ‡ means statistically significant difference over both strongest feature based baselines with $p < 0.01$ measured by the Student's paired t-test.

Methods	Accuracy	Precision	Recall	F1
Random Forest	0.4405	**0.6781**	0.4077	0.5092
AdaBoost	0.4430	0.5913	0.4187	0.4902
CNN-Context-Rep	**0.4708**‡	0.5647	**0.5129**	**0.5375**

Rep still achieves statistically significant improvement over both baselines in terms of accuracy with $p < 0.01$ measured by the Student's paired t-test.

3.6.4 Hyper-parameter Sensitivity Analysis

We further analyze the impact of two hyper-parameters on CNN-Context-Rep: the number of convolutional filters for the given utterance and the max sequence length. The choices for number of convolutional filters are (64, 128, 256, 512, 1024). We tune the max sequence length in (50, 100, 200, ..., 1000). As presented in Figure 3.6, the performance gradually increases as the number of filters increases. The best performance is at 1,024 filters. This confirms our expectation that more convolutional filters can extract richer features and thus produce better results. In addition, the performance fluctuates as the max sequence length increases. Performance with larger ($\geqslant 800$) max sequence length are better in general.

3.6.5 Case Study

Table 3.12 gives examples of the predictions that different systems fail to make. In the first utterance, the agent asks for the user's iOS version before providing a potential answer, which is a very common pattern of agents' responses. Our CNN-Context-Rep is able to identify the *Information Request* in the utterance while AdaBoost cannot. In the second utterance, both models fail to predict the *Negative Feedback*. This might be due to the fact that the feedback is not explicitly expressed.

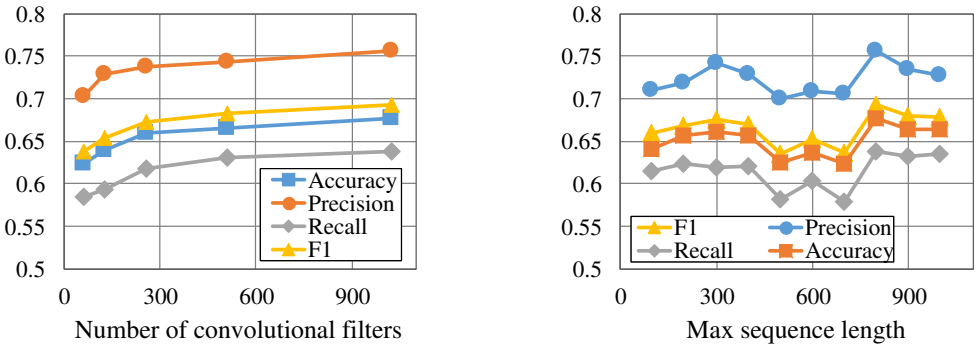


Figure 3.6: Performance of CNN-Context-Rep with different number of convolutional filters and max sequence length.

Table 3.12: Two utterances with their ground-truth and predicted user intent labels. Bold font indicates mispredicted content or labels. "Ours" refers to CNN-Context-Rep.

Hello. Welcome to Skype Community! **Please provide us the iOS version of your iPad**. The required iOS version for iPad is iOS 8 or higher and for the new Skype on iOS requires iOS 9 or higher. For more information, click here. Hope this helps. Let me know if you need further assistance. Thank you!			
Ground truth: **IR**, PA	Ours: **IR**, PA	AdaBoost: PA	Actor: agent
After modified the Windows entry, value of regedit, **the error also happened**. When I use C++ for creating another new Microsoft::Office::Interop::PowerPoint:: Application instance, the COMException is throwed.			
Ground truth: FD, **NF**	Ours: FD	AdaBoost: FD	Actor: user

In addition, it could be relevant that the number of feedback utterances in the training data is relatively limited compared to questions and answers, which makes it more difficult to predict positive/negative feedback.

3.7 Summary

In this chapter, we first create and annotate a large multi-turn question answering data for research in conversational search. We perform an in-depth characterization and analysis of this data to gain insights on the distribution, co-occurrence and flow pattern of user intent in information-seeking conversations. We then study two ap-

proaches to predict user intent in information-seeking conversations. First we use different ML methods with a rich feature set, including the content, structural, and sentiment features. We perform thorough feature importance analysis on both group level and individual level, which shows that structural features contribute most in this prediction task. Given findings from feature analysis, we construct enhanced neural classifiers to incorporate context information for user intent prediction. The enhanced neural model without feature engineering outperforms the baseline models by a large margin. Our findings can provide insights in the important factors of user intent prediction in information-seeking conversations.

CHAPTER 4

HISTORY MODELING FOR CONVERSATIONAL QUESTION ANSWERING

4.1 Introduction

In two recent ConvQA datasets, QuAC (Choi et al., 2019) and CoQA (Reddy et al., 2019), ConvQA is formalized as an answer span prediction problem similar in SQuAD (Rajpurkar et al., 2016, 2018). Specifically, given a question, a passage, and the conversation history preceding the question, the task is to predict a span in the passage that answers the question. In contrast to typical machine comprehension (MC) models, it is essential to make use of conversation history in this task. In this chapter, we introduce a general framework to deal with conversation history in ConvQA, where a history selection module first selects helpful history turns and a history modeling module then incorporates the selected turns. We then propose different implementations of the history selection and modeling modules in this framework.

On the aspect of history selection, existing models (Choi et al., 2019; Reddy et al., 2019) select conversation history with a simple heuristic that assumes immediate previous turns are more helpful than others. This assumption, however, is not necessarily true. Yatskar (2018) conducted a qualitative analysis on QuAC by observing 50 randomly sampled passages and their corresponding 302 questions. He showed that 35.4% and 5.6% of questions have the dialog behaviors of *topic shift* and *topic return* respectively. A topic shift suggests that the current question shifts to a new topic, such as the Q_3 in Table 1.1. While topic return means that the current question is about a topic that has previously been shifted away from. For example,

Q_7 returns to the same topic in Q_1 in Table 1.1. In both cases, the current question is not directly relevant to immediate previous turns. It could be unhelpful or even harmful to always incorporate immediate previous turns. Although we expect this heuristic to work well in many cases where the current question is *drilling down* on the topic being discussed, it might not work for topic shift or topic return. There is no published work that focuses on *learning* to select or re-weight conversation history turns. To address this issue, we propose a *history attention mechanism* (HAM) that learns to attend to all available history turns with different weights. This method increases the scope of candidate histories to include remote yet potentially helpful history turns. Meanwhile, it promotes useful history turns with large attention weights and demotes unhelpful ones with small weights. More importantly, the history attention weights provide explainable interpretations to understand the model results and thus can provide new insights in this task.

In addition, on the aspect of history modeling, some existing methods either simply prepend the selected history turns to the current question (Reddy et al., 2019; Zhu et al., 2018) or use complicated recurrent structures to model the conversation history (Huang et al., 2019), generating relatively large system overhead. We introduce a *history answer embedding* (HAE) method to incorporate the conversation history to BERT in a natural way. Moreover, since the utility of a history utterance could be related to its position, we propose to consider the position information in HAE, resulting in a *positional history answer embedding* (PosHAE) method. We show that position information plays an important role in conversation history modeling.

Furthermore, we introduce a new angle to tackle the problem of ConvQA. We take advantage of *multi-task learning* (MTL) to do answer span prediction along with another essential conversation task (dialog act prediction) using a uniform model architecture. Dialog act prediction is necessary in ConvQA systems because dialog acts can reveal crucial information about user intents and thus help the system pro-

vide better answers. More importantly, by applying this multi-task learning scheme, the model learns to produce more generic and expressive representations (Liu et al., 2019a), due to additional supervising signals and the regularization effect when optimizing for multiple tasks. We show that these benefits have contributions to the model performance for the dialog act prediction task.

In this work, we propose a novel solution to tackle ConvQA. We boost the performance from three different angles, i.e., history selection, history modeling, and multi-task learning. Our contributions can be summarized as follows:

1. To better conduct history selection, we introduce a history attention mechanism to conduct a "soft selection" for conversation histories. This method attends to history turns with different weights based on how helpful they are on answering the current question. This method enjoys good explainability and can provide new insights to the ConvQA task.

2. To enhance history modeling, we first propose a history answer embedding method and then incorporate the history position information into history answer embedding, resulting in a positional history answer embedding method. Inspired by the latest breakthrough in language modeling, we leverage BERT to jointly model the given question, passage and conversation history, where BERT is adapted to a conversation setting.

3. To further improve the performance of ConvQA, we jointly learn answer span prediction and dialog act prediction in a multi-task learning setting. We take advantage of MTL to learn more generalizable representations.

4. We conduct extensive experimental evaluations to demonstrate the effectiveness of our model and to provide new insights for the ConvQA task. The implementation of our model has been open-sourced to the research community.[1,2]

4.2 Conversational Question Answering

4.2.1 Task Definition

The ConvQA task is defined as follows (Choi et al., 2019; Reddy et al., 2019). Given a passage p, the k-th question q_k in a conversation, and the conversation history $\mathbf{H}_k$ preceding q_k, the task is to answer q_k by predicting an answer span a_k within the passage p. The conversation history $\mathbf{H}_k$ contains $k-1$ turns, where the i-th turn $\mathbf{H}_k^i$ contains a question q_i and its groundtruth answer a_i. Formally, $\mathbf{H}_k = \{(q_i, a_i)\}_{i=1}^{k-1}$. One of the unique challenges of ConvQA is to leverage the conversation history to understand and answer the current question.

Additionally, an important task relevant to conversation modeling is dialog act prediction. QuAC (Choi et al., 2019) provides two dialog acts, namely, *affirmation* (Yes/No) and *continuation* (Follow up). The affirmation dialog act v^a consists of three possible labels: {`yes, no, neither`}. The continuation dialog act v^c also consists of three possible labels: {`follow up, maybe follow up, don't follow up`}. Each question is labeled with both dialog acts. The labels for each dialog act are mutually exclusive. This dialog act prediction task is essentially two sentence classification tasks. Therefore, a complete training instance is composed of the model input $(q_k, p, \mathbf{H}_k)$ and its ground truth labels (a_k, v_k^a, v_k^c), where a_k and v_k^a, v_k^c are labels for answer span prediction and dialog act prediction respectively.

---

[1] `https://github.com/prdwb/attentive_history_selection`

[2] `https://github.com/prdwb/bert_hae`

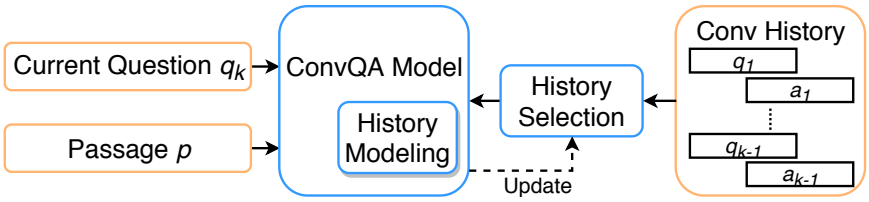


Figure 4.1: A general framework for ConvQA. Orange denotes model input and blue denotes model components.

4.2.2 A ConvQA Framework

We present an abstract framework for ConvQA with modularized design in Figure 4.1. It consists of three major components, a ConvQA model, a history selection module, and a history modeling module. In practice, the history modeling module can be a mechanism inside the ConvQA model. Given a training instance $(p, q_k, \mathbf{H}_k)$, the history selection module chooses a subset of the history turns $\mathbf{H}'_k$ that are expected to be more helpful than others. The history modeling module then incorporates $\mathbf{H}'_k$ into the ConvQA model. If the history selection module is a learned policy, the ConvQA model can generate a signal to guide its update. A variation of this framework is to first model all available conversation history and then make selections.

4.2.3 Model Overview

In the following sections, we present our model that tackles the two tasks described in Section 4.2.1 together.

Our proposed model consists of four components: an encoder, a history attention module, an answer span predictor, and a dialog act predictor. The encoder is a BERT model that encodes the question q_k, the passage p, and conversation histories $\mathbf{H}_k$ into contextualized representations. Then the history attention module learns to attend to history turns with different weights and computes aggregated representations for $(q_k, p, \mathbf{H}_k)$ on a token level and a sequence level. Finally, the two prediction modules

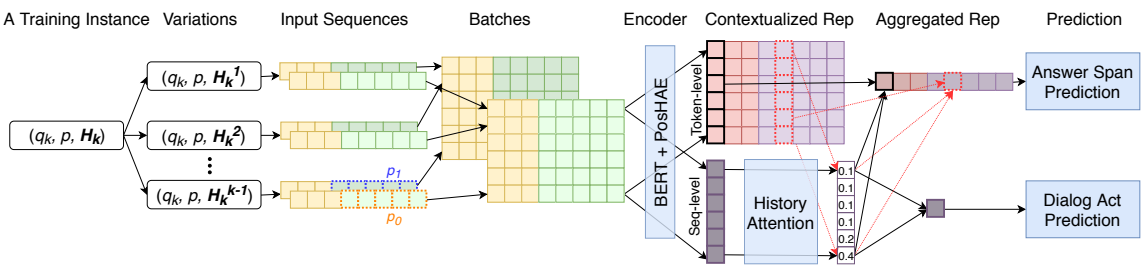


Figure 4.2: Our model consists of an encoder, a history attention module, an answer span predictor, and a dialog act predictor. Given a training instance, we first generate variations of this instance, where each variation contains the same question and passage, with only one turn of conversation history. We use a sliding window approach to split a long passage into "sub-passages" (p_0 and p_1) and use p_0 for illustration. The BERT encoder encodes the variations to contextualized representations on both token level and sequence level. The sequence-level representations are used to compute history attention weights. Alternatively, we propose a fine-grained history attention approach as marked in red-dotted lines. Finally, answer span prediction and dialog act predictions are conducted on the aggregated representations generated by the history attention module.

make predictions based on the aggregated representations with a multi-task learning setting.

In our architecture, history modeling is enabled in the BERT encoder, where we model one history turn at a time. History selection is performed in the history attention module in the form of "soft selection". Figure 4.2 gives an overview of our model. We illustrate each component in detail in the following sections.

4.2.4 Encoder

4.2.4.1 BERT Encoder

The encoder is a BERT model that encodes the question q_k, the passage p, and conversation histories $\mathbf{H}_k$ into contextualized representations. BERT is a pre-trained language model that is designed to learn deep bidirectional representations using transformers (Vaswani et al., 2017). Figure 4.3 gives an illustration of the encoder. It zooms in to the encoder component in Figure 4.2. It reveals the encoding process from an input sequence (the yellow-green row to the left of the encoder in Figure 4.2)

to a contextualized representation (the pink-purple row to the right of the encoder in Figure 4.2).

Given a training instance $(q_k, p, \mathbf{H}_k)$, we first generate $k-1$ variations of this instance, where each variation contains the same question and passage, with only one turn of conversation history. Formally, the i-th variation is denoted as $(q_k, p, \mathbf{H}_k^i)$, where $\mathbf{H}_k^i = (q_i, a_i)$. We follow the previous work (Devlin et al., 2019) and use a sliding window approach to split long passages, and thus construct multiple input sequences for a given instance variation. Suppose the passage is split into n pieces,[3] the training instance $(q_k, p, \mathbf{H}_k)$ would generate $n(k-1)$ input sequences. We take the $k-1$ input sequences corresponding to the first piece of the passage (still denoted as p here for simplicity) for illustration here. As shown in Figure 4.3, we pack the question q_k and the passage p into one sequence. The input sequences are fed into BERT and BERT generates contextualized token-level representations for each sequence based on the embeddings for tokens, segments, positions, and a special (positional) history answer embedding ((Pos)HAE). (Pos)HAE embeds the history answer a_i into the passage p since a_i is essentially a span of p. We describe this method in the next section.

The encoder can be formulated as a transformation function $F(\cdot)$ that takes in a training instance variation and produces a hidden representation for it on a token level, i.e., $\mathbf{T}_k^i = F(q_k, p, \mathbf{H}_k^i)$, where $\mathbf{T}_k^i \in \mathbb{R}^{M \times h}$ is the token-level representation for this instance variation. M is the sequence length, and h is the hidden size of the token representation. $\mathbf{T}_k^i$ can also be represented as $\{\mathbf{t}_k^i(m)\}_{m=1}^M$, where $\mathbf{t}_k^i(m) \in \mathbb{R}^h$ refers to the representation of the m-th token in $\mathbf{T}_k^i$. Instead of using separate encoders for questions, passages, and histories in previous work (Zhu et al., 2018; Huang et al.,

[3] $n = 2$ in Figure 4.2

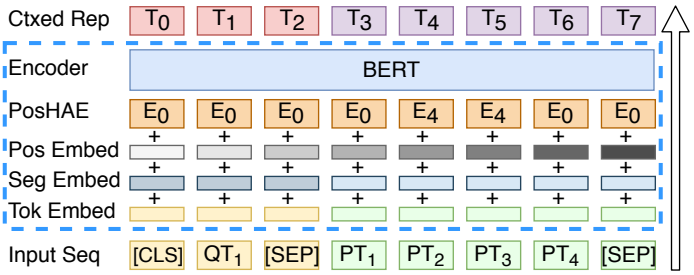


Figure 4.3: The encoder with PosHAE. It zooms in to the encoder in Fig. 4.2. It reveals the encoding process (marked by the blue-dotted lines) from an input sequence (the yellow-green row to the left of the encoder in Fig. 4.2) to contextualized representations (the pink-purple row to the right of the encoder in Fig. 4.2). QT_i/PT_i denote question/passage tokens. Suppose we are encoding $(q_6, p, \mathbf{H}_6^2)$, E_4 and E_0 are the history embeddings for tokens that are in and not in $\mathbf{H}_6^2$).

2019), we take advantage of BERT and PosHAE to model these different input types jointly.

In addition, we also obtain a sequence-level representation $\mathbf{s}_k^i \in \mathbb{R}^h$ for each sequence. We take the representation of the `[CLS]` token, which is the first token of the sequence, and pass it through a fully-connected layer that has h hidden units (Devlin et al., 2019). That is, $\mathbf{s}_k^i = \tanh(\mathbf{t}_k^i(1) \cdot \mathbf{W}_{\text{CLS}})$, where $\mathbf{W}_{\text{CLS}} \in \mathbb{R}^{h \times h}$ is the weight matrix for this dense layer. The bias term in this equation and following equations are omitted for simplicity. This is a standard technique to obtain a sequence-level representation in BERT. It is essentially a pooling method to remove the dimension of sequence length. We also conduct experiments with average pooling and max pooling on this dimension to achieve the same purpose.

4.2.4.2 History Answer Embedding

One important difference of MC and ConvQA lies in handling conversation history. Suppose we are given a subset of the conversation history chosen by the history selection module for the current question. There are various ways to model the selected history turns. The most intuitive way is to prepend the conversation history to the current question (Reddy et al., 2019; Zhu et al., 2018). In this work, we propose

a different approach to model the conversation history by giving tokens extra embedding information. Specifically, a *history answer embedding* (HAE) layer is included in addition to other embeddings. We learn two unique history answer embeddings that denote whether a token is part of history answers or not. This introduces the conversation history to BERT in a natural way. HAE modifies the embedding information for a token and thus has influence on the token representation generated by BERT, not only for this token but also for other tokens since BERT considers contextual information. This process also improves the prediction of the answer span as shown in the experiments. By representing conversation history with HAE, we turn an MC model into a ConvQA model.

4.2.4.3 Positional History Answer Embedding

A commonly used history selection method is to select immediate previous turns. The intuition is that the utility of a history utterance could be related to its position. Therefore, we further propose to consider the position information in HAE, resulting in a *positional history answer embedding* (PosHAE) method. The "position" refers to the relative position of a history turn in terms of the current question.

Specifically, we first define a vocabulary of size $I + 1$ for PosHAE, denoted as $V_{PosHAE} = \{0, 1, \ldots, I\}$, where I is the max number of history turns.[4] Given the current question q_k and a history turn H_k^i, we compute the relative position of H_k^i in terms of q_k as $k - i$. This relative position corresponds to a vocabulary ID in V_{PosHAE}. We use the vocabulary ID 0 for the tokens that are not in the given history. We then use a truncated normal distribution to initialize an embedding look up table $\mathbf{ET} \in \mathbb{R}^{|I+1| \times h}$. We use V_{PosHAE} to map each token to a history answer embedding in $\mathbf{ET}$. The history answer embeddings are learned. An example is illustrated in Figure 4.3. In addition to introducing conversation history, PosHAE enhances HAE

[4]In QuAC, $I = 11$, which means a dialog has at most 11 history turns.

by incorporating position information of history turns. This enables the ConvQA model to capture the spatial patterns of history answers in context.

4.2.5 History Attention Module

The core of the history attention module is a history attention mechanism (HAM). The inputs of this module are the token-level and sequence-level representations for all variations that are generated by the same training instance. The token-level representation is denoted as $\mathcal{T}_k = \{\mathbf{T}_k^i\}_{i=1}^I$, where $\mathcal{T}_k \in \mathbb{R}^{I \times M \times h}$. Similarly, the sequence-level representation is denoted as $\mathbf{S}_k = \{\mathbf{s}_k^i\}_{i=1}^I$, where $\mathbf{S}_k \in \mathbb{R}^{I \times h}$. The first dimension of $\mathcal{T}_k$ and $\mathbf{S}_k$ are both I because they are always padded to the max number of history turns. The padded parts are masked out. $\mathcal{T}_k$ and $\mathbf{S}_k$ are illustrated in Figure 4.2 as the "Token-level" and "Seq-level Contextualized Rep" respectively.

The history attention network is a single-layer feed-forward network. We learn an attention vector $\mathbf{D} \in \mathbb{R}^h$ to map a sentence representation $\mathbf{s}_k^i$ to a logit and use the softmax function to compute probabilities across all sequences generated by the same instance. Formally, the history attention weights are computed as follows.

$$w_i = \frac{e^{\mathbf{D} \cdot \mathbf{s}_k^i}}{\sum_{i'=1}^{I} e^{\mathbf{D} \cdot \mathbf{s}_k^{i'}}} \tag{4.1}$$

where w_i is the history attention weight for $\mathbf{s}_k^i$. Let $\mathbf{w} = \{w_i\}_{i=1}^I$. We compute aggregated representations for $\mathcal{T}_k$ and $\mathbf{S}_k$ with $\mathbf{w}$:

$$\hat{\mathbf{T}}_k = \sum_{i=1}^{I} \mathbf{T}_k^i \cdot w_i \quad , \qquad \hat{\mathbf{s}}_k = \sum_{i=1}^{I} \mathbf{s}_k^i \cdot w_i \tag{4.2}$$

where $\hat{\mathbf{T}}_k \in \mathbb{R}^{M \times h}$ and $\hat{\mathbf{s}}_k \in \mathbb{R}^h$ are aggregated token-level and sequence-level representations respectively. The attention weights $\{w_i\}_{i=1}^I$ are computed on a sequence-level and thus the tokens in the same sequence share the same weight. Intuitively, the history attention network attends to the variation representations with different weights

and then each variation representation contributes to the aggregated representation according to the utility of the history turn in this variation.

Alternatively, we develop a *fine-grained history attention* approach to compute the attention weights. Instead of using sequence-level representations $\mathbf{S}_k$ as the input for the attention network, we use the token-level ones. The token-level attention input for the m-th token in the sequence is denoted as $\mathbf{t}_k(m) = \{\mathbf{t}_k^i(m)\}_{i=1}^I$, where $\mathbf{t}_k(m) \in \mathbb{R}^{I \times h}$. This is marked as a column with red-dotted lines in Figure 4.2. Then these attention weights are applied to $\mathbf{t}_k(m)$ itself:

$$
\begin{aligned}
w_i &= \frac{e^{\mathbf{D} \cdot \mathbf{t}_k^i(m)}}{\sum_{i'=1}^I e^{\mathbf{D} \cdot \mathbf{t}_k^{i'}(m)}} \\
\hat{\mathbf{t}}_k(m) &= \sum_{i=1}^I \mathbf{t}_k^i(m) \cdot w_i
\end{aligned}
\tag{4.3}
$$

where $\hat{\mathbf{t}}_k(m) \in \mathbb{R}^h$ is the aggregated token representation for the m-th token in this sequence. Therefore, the aggregated token-level representation $\hat{\mathbf{T}}_k$ for this sequence is $\{\hat{\mathbf{t}}_k(m)\}_{m=1}^M$. We show the process of computing the aggregated token representation for one token, but the actual process is vectorized and paralleled for all tokens in this sequence. Intuitively, this approach computes the attention weights given different token representations for the same token but embedded with different history information. These attention weights are on a token level and thus are more fine-grained than those from the sequence-level representations.

In both granularity levels of history attention, we show the process of computing attention weights for a single instance, but the actual process is vectorized for multiple instances. Also, if the given question does not have history turns (i.e., the first question of a conversation), it should bypass the history attention module. In practice, this is equivalent to passing it though the history attention network since all the attention weights will be applied to itself.

4.2.6 Answer Span Prediction

Given the aggregated token-level representation $\hat{\mathbf{T}}_k$ produced by the history attention network, we predict answer span by computing the probability of each token being the begin token and the end token. Specifically, we learn two sets of parameters, a begin vector and an end vector, to map a token representation to a logit. Then we use the softmax function to compute probabilities across all tokens in this sequence. Formally, let $\mathbf{B} \in \mathbb{R}^h$ and $\mathbf{E} \in \mathbb{R}^h$ be the begin vector and the end vector respectively. The probabilities of this token being the begin token p_m^B and end token p_m^E are:

$$p_m^B = \frac{e^{\mathbf{B} \cdot \hat{\mathbf{t}}_k(m)}}{\sum_{m'=1}^{M} e^{\mathbf{B} \cdot \hat{\mathbf{t}}_k(m')}} \quad , \qquad p_m^E = \frac{e^{\mathbf{E} \cdot \hat{\mathbf{t}}_k(m)}}{\sum_{m'=1}^{M} e^{\mathbf{E} \cdot \hat{\mathbf{t}}_k(m')}} \tag{4.4}$$

We then compute the cross-entropy loss for answer span prediction:

$$\begin{gathered} \mathcal{L}_B = -\sum_{M} \mathbb{1}\{m = m_B\} \log p_m^B \quad , \quad \mathcal{L}_E = -\sum_{M} \mathbb{1}\{m = m_E\} \log p_m^E \\ \mathcal{L}_{ans} = \frac{1}{2}(\mathcal{L}_B + \mathcal{L}_E) \end{gathered} \tag{4.5}$$

where tokens at positions of m_B and m_E are the ground truth begin token and end token respectively, and $\mathbb{1}\{\cdot\}$ is an indicator function. $\mathcal{L}_B$ and $\mathcal{L}_E$ are the losses for the begin token and end token respectively and $\mathcal{L}_{ans}$ is the loss for answer span prediction. For unanswerable questions, a "CANNOTANSWER" token is appended to each passage in QuAC. The model learns to predict an answer span of this exact token if it believes the question is unanswerable.

Invalid predictions, including the cases where the predicted span overlaps with the question part of the sequence, or the end token comes before the begin token, are discarded at testing time.

4.2.7 Dialog Act Prediction

Given the aggregated sequence-level representation $\hat{\mathbf{s}}_k$ for a training instance, we learn two sets of parameters $\mathbf{A} \in \mathbb{R}^{|V_a| \times h}$ and $\mathbf{C} \in \mathbb{R}^{|V_c| \times h}$ to predict the dialog act of

affirmation and continuation respectively, where $|V_a|$ and $|V_c|$ denote the number of classes.[5] Formally, the loss for dialog act prediction for affirmation is:

$$
\begin{aligned}
p(v|\hat{\mathbf{s}}_k) &= \frac{e^{\mathbf{A}_v \cdot \hat{\mathbf{s}}_k}}{\sum_{v'=1}^{|V_a|} e^{\mathbf{A}_{v'} \cdot \hat{\mathbf{s}}_k}} \\
\mathcal{L}_A &= -\sum_{v} \mathbb{1}\{v = v_k^a\} \log p(v|\hat{\mathbf{s}}_k)
\end{aligned}
\tag{4.6}
$$

where $\mathbb{1}\{\cdot\}$ is an indicator function to show whether the predicted label v is the ground truth label v_k^a, and $\mathbf{A}_v \in \mathbb{R}^h$ is the vector in $\mathbf{A}$ corresponding to v. The loss $\mathcal{L}_C$ for predicting the continuation dialog act v_k^c is computed in the same way. We make dialog act predictions independently based on the information of each single training instance $(q_k, p, \mathbf{H}_k)$. We do not model history dialog acts in the encoder for this task.

4.2.8 Model Training

4.2.8.1 Batching

We implement an *instance-aware batching* approach to construct the batches for BERT. This method guarantees that the variations generated by the same training instance are always included in the same batch, so that the history attention module operates on all available histories. In practice, a passage in a training instance can produce multiple "sub-passages" (e.g., p_0 and p_1 in Figure 4.2) after applying the sliding window approach (Devlin et al., 2019). This results in multiple "sub-instances" (e.g. $(q_k, p_0, \mathbf{H}_k^i)$ and $(q_k, p_1, \mathbf{H}_k^i)$), which are modeled separately and potentially in different batches. This is because the "sub-passages" have overlaps to make sure that every passage token has sufficient context so that they can be considered as different passages.

[5] $|V_a| = 3$ and $|V_c| = 3$ in QuAC.

4.2.8.2 Training Loss and Multi-task Learning

We adopt the multi-task learning idea to jointly learn the answer span prediction task and the dialog act prediction task. Both tasks make predictions based on the aggregated representations produced by the history attention module. In other words, the history attention is supervised partially by the answer span and partially by the dialog act. All parameters are learned in an end-to-end manner. We use hyper-parameters λ and μ to combine the losses for different tasks. That is,

$$\mathcal{L} = \mu\mathcal{L}_{ans} + \lambda\mathcal{L}_A + \lambda\mathcal{L}_C \tag{4.7}$$

where $\mathcal{L}$ is the total training loss.

Multi-task learning has been shown to be effective for representation learning (Liu et al., 2019a, 2015; Xu et al., 2018). There are two reasons behind this. 1) Our two tasks provide more supervising signals to fine-tune the encoder. 2) Representation learning benefits from a regularization effect by optimizing for multiple tasks. Although BERT serves as a universal encoder by pre-training with a large amount of unlabeled data, MTL is a complementing technology (Liu et al., 2019a) that makes such representations more generic and transferable. More importantly, we can handle two essential tasks in ConvQA, answer span prediction and dialog act prediction, with a uniform model architecture.

4.3 Experiments

4.3.1 Data Description

We experiment with the QuAC (Question Answering in Context) dataset (Choi et al., 2019). It is a large-scale dataset designed for modeling and understanding information-seeking conversations. It contains interactive dialogs between an information-seeker and an information-provider. The information-seeker tries to learn

Table 4.1: Data statistics. We can only access the training and validation data.

Items	Train	Validation
# Dialogs	11,567	1,000
# Questions	83,568	7,354
# Average Tokens Per Passage	396.8	440.0
# Average Tokens Per Question	6.5	6.5
# Average Tokens Per Answer	15.1	12.3
# Average Questions Per Dialog	7.2	7.4
# Min/Avg/Med/Max History Turns Per Question	0/3.4/3/11	0/3.5/3/11

about a *hidden* Wikipedia passage by asking a sequence of freeform questions. She/he only has access to the heading of the passage, simulating an information need. The information-provider answers each question by providing a short span of the given passage. One of the unique properties that distinguish QuAC from other dialog data is that it comes with dialog acts. The information-provider uses dialog acts to provide the seeker with feedback (e.g., "ask a follow up question"), which makes the dialogs more productive (Choi et al., 2019). This dataset poses unique challenges because its questions are more open-ended, sometimes unanswerable, or only meaningful within the dialog context. More importantly, many questions have coreferences and interactions with conversation history, making this dataset suitable for our task. We present some statistics of the dataset in Table 4.1.

4.3.2 Experimental Setup

4.3.2.1 Competing Methods

We consider all methods with published papers on the QuAC leaderboard at the time as baselines.[6] To be specific, the competing methods are:

[6]The methods without published papers or descriptions are essentially done in parallel with ours and may not be suitable for comparison since their model details are unknown. Besides, these works could be using generic performance boosters, such as BERT-large, data augmentation, transfer learning, or better training infrastructures.

- **BiDAF++** (Peters et al., 2018; Choi et al., 2019): BiDAF (Seo et al., 2016) is a top-performing SQuAD model. It uses bi-directional attention flow mechanism to obtain a query-aware context representation. BiDAF++ makes further augmentations with self-attention (Gardner and Clark, 2018) and contextualized embeddings.

- **BiDAF++ w/ 2-Context** (Choi et al., 2019): This model incorporates conversation history by modifying the passage and question embedding processes. Specifically, it encodes the dialog turn number with the question embedding and concatenates answer marker embeddings to the word embedding.

- **FlowQA** (Huang et al., 2019): This model incorporates conversation history by integrating intermediate representation generated when answering the previous question. Thus it is able to grasp the latent semantics of the conversation history compared to shallow approaches that concatenate history turns.

- **BERT**: This is a BERT based machine comprehension model following Devlin et al. (2019).

- **BERT + Prepend History Turns**: On top of BERT, we consider conversation history by prepending history turn(s) to the current question. **BERT + PHQA** prepends both history questions and answers; **BERT + PHA** prepends history answers only.

- **BERT + HAE**: On top of BERT, we use HAE to enable a seamless integration of conversation history. This is designed to test the performance of HAE. We set the max number of history turns to 6 since it gives the best performance under this setting.

- **BERT + PosHAE**: We enhance the BERT + HAE model with the PosHAE that we proposed. This method considers the position information of history

turns and serves as a stronger baseline. We also set the max number of history turns to 6.

- **HAM** (History Attention Mechanism): This is the solution we proposed in Section 4.2. It employs PosHAE for history modeling, the history attention mechanism for history selection, and the MTL scheme to optimize for both answer span prediction and dialog act prediction tasks. We use the fine-grained history attention in Equation 4.3. We use "HAM" as the model name since the attentive history selection is the most important and effective component that essentially defines the model architecture.

- **HAM (BERT-Large)**: Due to the competing nature of the QuAC challenge, we apply BERT-Large to HAM for a more informative evaluation. This is more resource intensive. Other HAM models in this chapter are constructed with BERT-Base for two reasons: 1) To alleviate the memory and training efficiency issues caused by BERT-Large and thus speed up the experiments for the research purpose. 2) To keep the settings consistent with other baselines for fair and easy comparison.

4.3.2.2 Evaluation Metrics

The QuAC challenge provides two evaluation metrics, the word-level F1, and the human equivalence score (HEQ) (Choi et al., 2019). The word-level F1 evaluates the overlap of the prediction and the ground truth answer span. It is a classic metric used in MC and ConvQA tasks (Rajpurkar et al., 2016; Reddy et al., 2019; Choi et al., 2019). HEQ measures the percentage of examples for which system F1 exceeds or matches human F1. Intuitively, this metric judges whether a system can provide answers as good as an average human. This metric is computed on the question level (HEQ-Q) and the dialog level (HEQ-D). In addition, the dialog act prediction task is evaluated by accuracy.

4.3.2.3 Implementation Details

Models are implemented in TensorFlow[7] based on the SQuAD model on BERT.[8] The version of the QuAC data we use is v0.2. We use the BERT-Base Uncased model with the max sequence length set to 384. We train the ConvQA model with a Adam weight decay optimizer with an initial learning rate of 3e-5. The warming up portion for learning rate is 10%. We set the stride in the sliding window for passages to 128 and the max question length to 64. For BERT + PHQA/PHA/HAE/PosHAE, the batch size is set to 12. For HAM, The batch size is 24. The total training step is set to 30,000. Experiments are conducted on a single NVIDIA TESLA M40 GPU. λ and μ for multi-task learning is set to 0.1 and 0.8 respectively for HAM.

4.3.3 Main Evaluation Results

We report the results on the validation and test sets in Table 4.2. Our best model was evaluated officially by the QuAC challenge and the result is displayed on the leaderboard.[9] Since dialog act prediction is not the main task of this dataset, most of the baseline methods do not perform this task.

We summarize our observations of the results as follows.

1. Incorporating conversation history significantly boosts the performance in ConvQA. This is true for both BiDAF++ and BERT-based models. This suggests the importance of conversation history in the ConvQA task.

2. Our BERT-based ConvQA model outperforms BiDAF++. Furthermore, BERT with any of the history modeling methods outperform BiDAF++ w/ 2-Context. This shows the advantage of using BERT for ConvQA.

---

[7] https://www.tensorflow.org/

[8] https://github.com/google-research/bert/blob/master/run_squad.py

[9] http://quac.ai/

Table 4.2: Evaluation results on QuAC. Models in a bold font are our implementations. Each cell displays val/test scores. Val result of BiDAF++, FlowQA are from Choi et al. (2019) and Huang et al. (2019). Test results are from the QuAC leaderboard. ‡ means statistically significant improvement over the strongest baseline with $p < 0.05$ tested by the Student's paired t-test. We can only do significance test on F1 on the validation set. "–" means a result is not available and "N/A" means a result is not applicable for this model. We set the max answer length to 30 wordpieces for the second group and 40 for the third group.

Models	F1	HEQ-Q	HEQ-D	Yes/No	Follow up
BiDAF++	51.8 / 50.2	45.3 / 43.3	2.0 / 2.2	86.4 / 85.4	59.7 / 59.0
BiDAF++ w/ 2-C	60.6 / 60.1	55.7 / 54.8	5.3 / 4.0	86.6 / 85.7	61.6 / 61.3
FlowQA	64.6 / 64.1	– / 59.6	– / 5.8	N/A	N/A
BERT	54.4 / –	48.9 / –	2.9 / –	N/A	N/A
BERT + PHQA	62.0 / –	57.5 / –	5.4 / –	N/A	N/A
BERT + PHA	61.8 / –	57.5 / –	4.7 / –	N/A	N/A
BERT + HAE	63.1 / 62.4	58.6 / 57.8	6.0 / 5.1	N/A	N/A
BERT + HAE	63.9 / –	59.7 / –	5.9 / –	N/A	N/A
BERT + PosHAE	64.7 / –	60.7 / –	6.0 / –	N/A	N/A
HAM	65.7‡ / 64.4	62.1 / 60.2	7.3 / 6.1	**88.3** / **88.4**	62.3 / **61.7**
HAM (Large)	**66.7**‡ / **65.4**	**63.3** / **61.8**	**9.5** / **6.7**	88.2 / 88.2	**62.4** / 61.0

3. Prepending history turns with PHQA and PHA are both effective. The fact that they achieve similar performance suggests that history questions contribute little to the performance. This verifies our observation of the data that most follow-up questions are relevant to history answers.

4. Our HAE approach achieves better performance than simply prepending history turns. This indicates that HAE is more effective in modeling conversation history when the length of the input sequence is limited.

5. BERT + PosHAE brings a significant improvement compared with BERT + HAE, achieving the best results among baselines. This suggests that the position information plays an important role in conversation history modeling with history answer embedding. This shows the effectiveness of this conceptually simple idea of modeling conversation history in BERT with PosHAE.

6. Our model HAM obtains statistically significant improvements over BERT + PosHAE, the strongest baseline, with $p < 0.05$ tested by the Student's paired t-test. These results demonstrate the effectiveness of our method.

7. Our model HAM also achieves a substantially higher performance on dialog act prediction compared to baseline methods, showing the strength of our model on both tasks. We can only do significance test on F1. We are unable to do a significance test on dialog act prediction because the prediction results of BiDAF++ is not available. In addition, the sequence-level representations of HAM are obtained with max pooling. We see no major differences when using different pooling methods.

8. Applying BERT-Large to HAM brings a substantial improvement to answer span prediction, suggesting that a more powerful encoder can boost the performance.

4.3.4 Ablation Analysis

Section 4.3.3 shows the effectiveness of our model. This performance is closely related to several design choices. So we conduct an ablation analysis to investigate the contributions of each design choice by removing or replacing the corresponding component in the complete HAM model. Specifically, we have four settings as follows.

- **HAM w/o Fine-grained (F-g) History attention**. We use the sequence-level history attention (Equation 4.1 and 4.2) instead of the fine-grained history attention (Equation 4.3).

- **HAM w/o History Attention**. We do not learn any form of history attention. Instead, we override the history attention module to always produce equal weights. Note that this is not equivalent to "BERT + PosHAE". "BERT + PosHAE" incorporates the selected history turns in a single input sequence and relies on the

encoder to work out the importance of these history turns. The architecture we illustrated in Figure 4.2 models each history turn separately and capture their importance by the history attention mechanism explicitly, which is a more direct and explainable way. Therefore, even when we disable the history attention module, it is not equivalent to "BERT + PosHAE".

- **HAM w/o PosHAE**. We use HAE instead of the PosHAE we proposed in Section 4.2.4.3.

- **HAM w/o MTL**. Our multi-task learning scheme consists of two tasks, an answer span prediction task and a dialog act prediction task. Therefore, to evaluate the contribution of MTL, we further design two settings: (1) In **HAM w/o Dialog Act Prediction**, we set $\mu = 1$ and $\lambda = 0$ in Equation 4.7 to block the parameter updates from dialog act prediction. (2) In **HAM w/o Answer Span Prediction**, we set $\mu = 0$ in Equation 4.7 and thus block the updates caused by answer span prediction. We tune λ in (0.2, 0.4, 0.6, 0.8) in Equation 4.7 and try different pooling methods to obtain the sequence-level representations. We finally adopt $\lambda = 0.2$ and average pooling since they give the best performance. We consider these two ablation settings to fully control the factors in our experiments and thus precisely capture the differences in the representation learning caused by different tasks.

The ablation results on the validation set are presented in Table 4.3. The following are our observations.

1. By replacing the fine-grained history attention with sequence-level history attention, we observe a performance drop. This shows the effectiveness of computing history attention weights on a token level. This is intuitive because these weights are specifically tailored for the given token and thus can better capture the history information embedded in the token representations.

Table 4.3: Results for ablation analysis. These results are obtained on the validation set since the test set is hidden for official evaluation only. "w/o" means to remove or replace the corresponding component. † means statistically significant performance *decrease* compared to the complete HAM model with $p < 0.05$ tested by the Student's paired t-test. We can only do significance test on F1 and dialog act accuracy.

Models	F1	HEQ-Q	HEQ-D	Yes/No	Follow up
HAM	65.7	62.1	7.3	88.3	62.3
w/o F-g History Attention	64.9†	61.0	7.1	88.4	62.1
w/o History Attention	61.1†	57.2	6.4	87.9	60.5†
w/o PosHAE	64.2†	60.0	7.3	88.6	62.1
w/o Dialog Act Prediction	65.9	62.2	8.2	N/A	N/A
w/o Answer Span Prediction	N/A	N/A	N/A	86.2†	59.7†

2. When we disable the history attention module, we notice the performance drops dramatically by 4.6% and 3.8% compared with HAM and "HAM w/o F-g History Attention" respectively. This indicates that the history attention mechanism, regardless of granularity, can attend to conversation histories according to their importance. Disabling history attention also hurts the performance for dialog act prediction.
3. Replacing PosHAE with HAE also witnesses a major drop in model performance. This again shows the importance of history position information in modeling conversation history.
4. When we remove the dialog act prediction task, we observe that the performance for answer span prediction has a slight and insignificant increase. This suggests that dialog act prediction does not contribute to the representation learning for answer span prediction. Since dialog act prediction is a secondary task in our setting, its loss is scaled down and thus could have a limited impact on the optimization for the encoder. Although the performance for our main model is slightly lower on answer span prediction, it can handle both answer span prediction and dialog prediction tasks in a uniform way.

5. On the contrary, when we remove the answer span prediction task, we observe a relatively large performance drop for dialog act prediction. This indicates that the additional supervising signals from answer span prediction can indeed help the encoder to produce a more generic representation that benefits the dialog act prediction task. In addition, the encoder could also benefit from a regularization effect because it is optimized for two different tasks, and thus, alleviates overfitting. Although the multi-task learning scheme does not contribute to answer span prediction, we show that it is beneficial to dialog act prediction.

4.3.5 Case Study and Attention Visualization

One of the major advantages of our model is its explainability of history attention. In this section, we present a case study that visualizes the history attention weights predicted by our model.

In Chapter 3, we observe that *follow up questions* is one of the most important user intents in information-seeking conversations. Yatskar (2018) further described three history-related dialog behaviors that can be considered as a fine-grained taxonomy of follow up questions. We use these definitions to interpret the attention weights. These dialog behaviors are as follow.

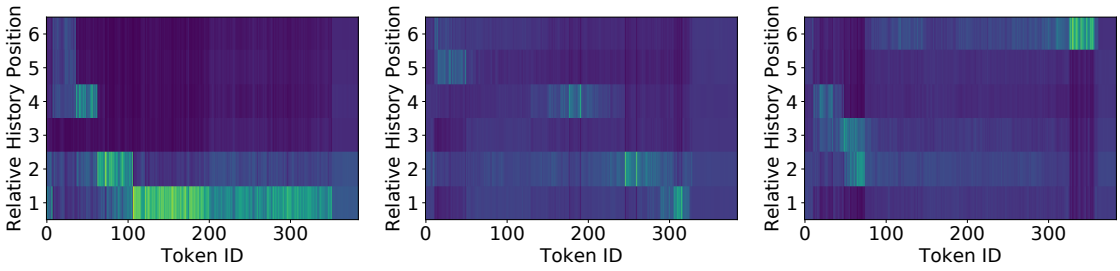
- **Drill down**: the current question is a request for more information about a topic being discussed.
- **Topic shift**: the current question is not immediately relevant to something previously discussed.
- **Topic return**: the current question is asking about a topic again after it had previously been shifted away from.

We keep records of the attention weights generated at testing time on the validation data. We use a sliding window approach to split long passages as mentioned

in Section 4.2.4.1. However, we specifically choose short passages that can be put in a single input sequence for easier visualization. The attention weights obtained from our fine-grained history attention model are visualized in Figure 4.4 and the corresponding dialogs are presented in Table 4.4.

Our history attention weights are computed on the token level. We observe that salient tokens are typically in the corresponding history answer in the passage. This suggests that our model learns to attend to tokens that carry history information. These tokens also bring some attention weights to other tokens that are not in the history answer since the token representations are contextualized. Although each history turn has an answer, the weights vary to reflect the importance of the history information.

We further interpret the attention weights with examples for different dialog behaviors. First, Table 4.4a shows that the current question is drilling down on more relevant information on the topic being discussed. In this case, the current question is closely related to its immediate previous turns. We observe in Figure 4.4a that our model can attend to these turns properly with greater weights assigned to the most immediate previous turn. Second, in the topic shift scenario presented in Table 4.4b and Figure 4.4b, the current question is not immediately relevant to its preceding history turns. Therefore, the attention weights are distributed relatively evenly across history turns. Third, as shown in Table 4.4c and Figure 4.4c, the first turn talks about the topic of musical career while the following turns shift away from this topic. The information-seeker returns to musical career in the current turn. In this case, the most important history turn to consider is the most remote one from the current question. Our model learns to attend to certain tokens in the first turn with larger weights, suggesting that the model could capture the topic return phenomenon. Moreover, we observe that the model does not attend to the passage token of "CANNOTANSWER", further indicating that it can identify useful history answers.



(a) Drill down (b) Topic shift (c) Topic return

Figure 4.4: Attention visualization for different dialog behaviors. Brighter spots mean higher attention weights. Token ID refers to the token position in an input sequence. A sequence contains 384 tokens. Relative history position refers to the difference of the current turn # with a history turn #. The selected examples are all in the 7th turn. These figures are best viewed in color.

4.4 Summary

In this chapter, we propose a novel model for ConvQA. We introduce a history attention mechanism to conduct a "soft selection" for conversation histories. We show that our model can capture the utility of history turns. In addition, we propose the history answer embedding method and enhance it by incorporating the position information for history turns. We show that history position information plays an important role in conversation history modeling. Finally, we propose to jointly learn answer span prediction and dialog act prediction with a uniform model architecture in a multi-task learning setting. Our extensive experimental evaluations have demonstrated the effectiveness of our model.

Table 4.4: QuAC dialogs that correspond to the dialog behaviors in Fig. 4.4. The examples are all in the 7th turn. "#" refers to the relative history position, which means "0" is the current turn and "6" is the most remote turn from the current turn. Each turn has a question and an answer, with the answer in italic. Co-references and related terms are marked in the same color.

(a) Drill down

#	Utterance
6	When did Ride leave NASA? *In 1987, Ride left ... to work at ...*
5	What did she do at ...? *International Security ...*
4	How long was she there? *In 1989, she became a professor ...*
3	Was she successful as a professor? *CANNOTANSWER*
2	Did she have any other professions? *Ride led two ... programs for ...*
1	What was involved in the programs? *The programs allowed ...*
0	What did she do after this? *To be predicted ...*

(b) Topic shift

#	Utterance
6	When did the Greatest Hits come out *beginning of 2004*
5	What songs were on the album *... "I Promised Myself" ...*
4	Was the album popular *... became another top-two hit ...*
3	Did it win any awards *CANNOTANSWER*
2	Why did they release this *... released in selected European ...*
1	Did they tour with this album? *the band finished their tour*
0	... interesting aspects about this article? *To be predicted ...*

(c) Topic return

#	Utterance
6	What is ... about Lorrie's musical career? *... she signed with ... her first album ...*
5	What songs are included in the album? *CANNOTANSWER*
4	... interesting aspects about this article? *made her first appearance ... at age 13,*
3	What did she do after her first appearance? *... she ... began leading the group ...*
2	What important work did she do ...? *leading the group through ...*
1	What songs did she played with the group? *CANNOTANSWER*
0	... interesting aspects of her musical career? *To be predicted ...*

CHAPTER 5

HISTORY MODELING FOR OPEN-RETRIEVAL CONVERSATIONAL QUESTION ANSWERING

5.1 Introduction

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 5.1.

1. **Open-retrieval**. This is a defining property of ORConvQA. In recent ConvQA datasets (Choi et al., 2019; Reddy et al., 2019), 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

Table 5.1: 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 (Lee et al., 2019b; Das et al., 2019)	✓	✗	-	-
Response Ranking w/ UDC (Lowe et al., 2015; Yang et al., 2018)	✗	✓	✓	✓
Conversational MC w/ CoQA (Zhu et al., 2018; Chen et al., 2019a)	✗	✓	-	✗
Conversational MC w/ QuAC (Huang et al., 2019)	✗	✓	✓	✓
ORConvQA w/ OR-QuAC (this chapter)	✓	✓	✓	✓

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 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 (Rajpurkar et al., 2016), 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" (Ahmad et al., 2019a) and have been reported to have more lexical overlap with their answers in SQuAD (Ahmad et al., 2019a). 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 5.1. We address this issue by aggregating existing data to create the *OR-QuAC* dataset. The QuAC (Choi et al., 2019) 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 (Elgohary et al., 2019) 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 have released OR-QuAC to the community to facilitate research on ORConvQA.[1]

In addition to proposing ORConvQA and creating the OR-QuAC dataset, we develop a system for ORConvQA following previous work on open-retrieval QA (Lee et al., 2019b). Our end-to-end system features a retriever, a reranker, and a reader that are all based on Transformers (Vaswani et al., 2017). We enable history modeling in all components by concatenating history questions to the current question. The passage retriever first retrieves the top K 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 (Ahmad et al., 2019a; Lee et al., 2019b; Das et al., 2019; Karpukhin et al., 2020; Xiong et al., 2020; Luan et al., 2020; Guu et al., 2020) that uses separate ALBERT (Lan et al., 2019) 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 (Devlin et al., 2019) encoder. It encodes the input sequence of a concatenation of the question, history, and each retrieved passage to contextualized representations for reranking and answer extraction. We incorporate shared-normalization (Gardner and Clark, 2018) 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

[1] `https://ciir.cs.umass.edu/downloads/ORConvQA/`

with the question encoder for an efficient retrieval with maximum inner product search (MIPS) (Shrivastava and Li, 2014; Johnson et al., 2019). The top retrieved passages are fed to the reranker and reader for a concurrent learning of all model components.

The open-retrieval setting presents challenges to training the ConvQA system. We study both *full supervision* and *weak supervision* approaches for training. In the fully-supervised setting, we encourage the model to find *the gold passage that comes with the dataset* and extract an answer from it by manually including the gold passage in the retrieval results during training. 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 (Yang et al., 2019b) 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.

This full supervision approach discussed above may not be sufficient for some real-world applications since gold passages are not always be available. Therefore, we further conduct study on weak supervision approaches for training a ORConvQA system. Within the scope of weak supervision, previous work (Chen et al., 2017; Lee et al., 2019b; Das et al., 2019) identify weak answers in the retrieval results by finding a span that is an exact match to the known answer. We argue that the effectiveness of this *span-match weak supervision* approach is contingent on having only *span answers* that are either short, or extractive spans of a retrieved passage. In information-seeking conversations, however, answers can be relatively long and are

not necessarily strict spans of any passage. These *freeform answers* can be challenging to handle for span-match weak supervision.

Therefore we introduce a *learned weak supervision* approach that can identify a paraphrased span of the known answer in a retrieved passage as the weak answer. Our method is more flexible than span-match weak supervision since it can handle both span answers and freeform answers. Moreover, our method is less demanding on the retriever because it can discover weak answers even when the retriever fails to retrieve any passage that contains an exact match of the known answer. By using a weakly-supervised training approach, our ConvQA system can discover answers in passages beyond the gold ones and thus can potentially leverage various knowledge sources. In other words, our learned weak supervision approach makes it possible for an ORConvQA system to be trained on natural conversations that can have long and freeform answers. The choice of the passage collection is no longer a part of the task definition. We can potentially combine different knowledge sources with these conversations since the weak answers can be discovered automatically.

Our learned weak supervisor is based on Transformers (Vaswani et al., 2017). Due to the lack of training data to learn this module, we propose a novel training method for the learned weak supervisor by leveraging a diverse paraphraser (Krishna et al., 2020) to generate the training data. Once the learned weak supervisor is trained, it is frozen and used to facilitate the training of the ORConvQA model.

In addition to continuing using OR-QuAC as an example for span answers, we further extend CoQA (Reddy et al., 2019) to an open-retrieval setting to evaluate our approach on freeform answers. We show that although a span-match weak supervisor can handle conversations with span answers, it is not sufficient for those with freeform answers. For more natural conversations with freeform answers, we demonstrate that our learned weak supervisor can outperform span match, proving the capability of our method in dealing with freeform answers. Moreover, by combining our learned weak

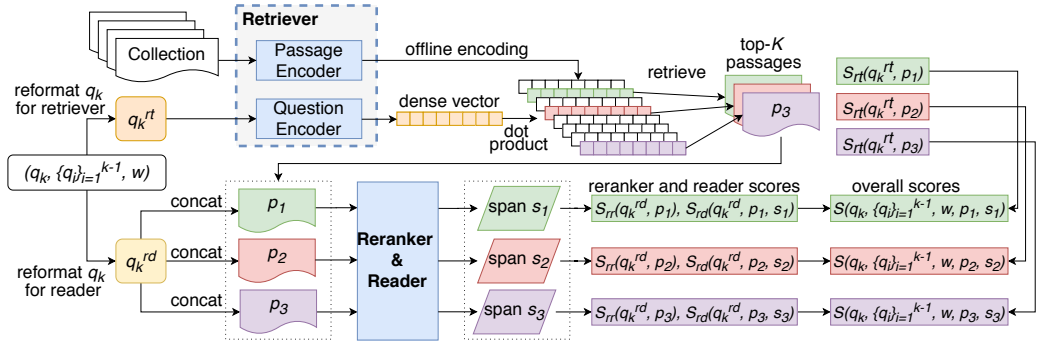


Figure 5.1: Architecture of our end-to-end ORConvQA model. The input is the current question q_k, all history questions $\{q_i\}_{i=1}^{k-1}$, and a history window size w. The retriever first retrieves top-K passages from the collection and generates retriever scores S_{rt}. The reranker and reader then 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.

supervisor with span match, the system has a significant improvement over using any one of the methods alone, indicating these two methods complement each other. Finally, we perform in-depth quantitative and qualitative analyses to provide more insight into weakly-supervised ORConvQA. Our data and model implementations are available for research purposes.[2,3]

5.2 Fully-supervised Open-retrieval Conversational QA

5.2.1 Our Approach

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.

---

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

[3] https://github.com/prdwb/ws-orconvqa

5.2.1.1 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.

Extractive models are trained on the supervision signals of the position of a span in the gold passage. In this section, we adopt a fully-supervised setting as the first step: 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 further present our work on weak supervision methods in Section 5.3.

5.2.1.2 Model Overview

We now present an end-to-end system that deals with the ORConvQA task described in Section 5.2.1.1. 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 5.1, the passage retriever first retrieves top-K 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.

5.2.1.3 Passage Retriever

We present the retriever module in the upper-left part of Figure 5.1. We follow previous research (Ahmad et al., 2019a; Lee et al., 2019b; Das et al., 2019) 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 (Lan et al., 2019) models for both encoders. ALBERT is a lite BERT (Devlin et al., 2019) model for learning bidirectional language representations from Transformers (Vaswani et al., 2017). It reduces the parameters of BERT by cross-layer parameter sharing and embedding parameters factorization (Lan et al., 2019).

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 5.2.2.1. 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 (Devlin et al., 2019). We then take the `[CLS]` representation and project it to a 128-dimensional vector as the question representation following Lee et al. (2019b). Formally,

$$v_q = W_q \text{ ALBERT}_q(q_k^{rt})\texttt{[CLS]} \tag{5.1}$$

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:

$$v_p = W_p \text{ ALBERT}_p(p_j)\texttt{[CLS]} \tag{5.2}$$

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

$$S_{rt}(q_k^{rt}, p_j) = v_q v_p^\top \tag{5.3}$$

5.2.1.4 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. (2019b) and Yang et al. (2019b), we introduce reranking into this process with little additional cost. Our reader mostly follows the standard architecture of a BERT based MC model (Devlin et al., 2019). We enhance this model by applying the shared-normalization mechanism proposed by Gardner and Clark (2018) to enable comparison across all retrieved passages for a question. Similar mechanisms are also adopted by Lee et al. (2019b).

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

$$v_{\texttt{[m]}} = \text{BERT}((q_k^{rd},\ p_j))_{\texttt{[m]}} \tag{5.4}$$

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

$$v_{\texttt{[CLS]}} = W_{\texttt{[CLS]}}\ \text{BERT}((q_k^{rd},\ p_j))\,\texttt{[CLS]} \tag{5.5}$$

where $W_{\texttt{[CLS]}}$ is a projection for the `[CLS]` representation to obtain the sequence representation $v_{\texttt{[CLS]}}$ following Devlin et al. (2019).

Reranker. As shown in Figure 5.1. 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 $v_{\texttt{[CLS]}}$ to a reranking score S_{rr}:

$$S_{rr}(q_k^{rd}, p_j) = \mathbf{W}_{rr} v_{\texttt{[CLS]}} \tag{5.6}$$

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:

$$S_s(q_k^{rd}, p_j, \texttt{[m]}) = \mathbf{W}_s v_{\texttt{[m]}} \quad , \quad S_e(q_k^{rd}, p_j, \texttt{[m]}) = \mathbf{W}_e v_{\texttt{[m]}} \tag{5.7}$$

where $S_s(q_k^{rd}, p_j, \texttt{[m]})$ and $S_e(q_k^{rd}, p_j, \texttt{[m]})$ 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 5.2.1.6.

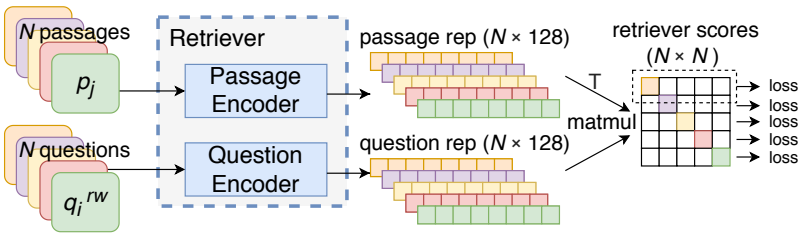


Figure 5.2: Retriever pretraining.

5.2.1.5 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.

Retriever Pretraining. We follow previous work (Lee et al., 2019b) to pretrain the retriever so that it gives a reasonable performance in the concurrent learning phase.

In Section 5.2.1.3, 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 5.2. 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

$$\mathbf{S}_{rt}(V_q, V_p) = V_q V_p^\top \tag{5.8}$$

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:

$$P_{rt}(p_j|q_i^{rw}) = \frac{\exp\left(S_{rt}(q_i^{rw}, p_j)\right)}{\sum_{j'=1}^{N} \exp\left(S_{rt}(q_i^{rw}, p_{j'})\right)} \tag{5.9}$$

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.

$$\mathcal{L}_{\text{pretrain}} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbb{1}\{j = i\} \log P_{rt}(p_j|q_i^{rw}) \tag{5.10}$$

Lee et al. (2019b) 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[4], 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.

Concurrent Learning of the Retriever, Reranker, and Reader. As indicated in Figure 5.1, 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 5.1. 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

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

the top K_{rt} passages, where $K_{rt} > K_{rd}$, for an aggressive update of the retriever following Lee et al. (2019b). 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.

$$\mathcal{L}_{rt} = -\sum_{p_j \in TK_{rt}} \mathbb{1}\{j = j_{rt}\} \log P_{rt}(p_j|q_k^{rt}) \tag{5.11}$$

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 5.6. We then compute the reranking probability and the reranking loss as follows.

$$P_{rr}(p_j|q_k^{rd}) = \frac{\exp\left(S_{rr}(q_k^{rd}, p_j)\right)}{\sum_{p_{j'} \in TK_{rd}} \exp\left(S_{rr}(q_k^{rd}, p_{j'})\right)} \tag{5.12}$$

$$\mathcal{L}_{rr} = -\sum_{p_j \in TK_{rd}} \mathbb{1}\{j = j_{rd}\} \log P_{rr}(p_j|q_k^{rd}) \tag{5.13}$$

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 (Gardner and Clark, 2018) 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.

$$P_s(q_k^{rd}, p_j, \texttt{[m]}) = \frac{\exp\left(S_s(q_k^{rd}, p_j, \texttt{[m]})\right)}{\sum_{p_{j'} \in TK_{rd}} \sum_{\texttt{[m']} \in (q_k^{rd},\ p_{j'})} \exp\left(S_s(q_k^{rd}, p_{j'}, \texttt{[m']})\right)} \tag{5.14}$$

$$\mathcal{L}_s = -\sum_{p_j \in TK_{rd}} \sum_{\texttt{[m]} \in (q_k^{rd},\ p_j)} \mathbb{1}\{j = j_{rd}, \texttt{[m]} = \texttt{[S]}\} \log P_s(q_k^{rd}, p_j, \texttt{[m]}) \tag{5.15}$$

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.

$$\mathcal{L}_{rd} = \frac{1}{2}(\mathcal{L}_s + \mathcal{L}_e) \tag{5.16}$$

Finally, the concurrent learning loss is computed as:

$$\mathcal{L} = \mathcal{L}_{rt} + \mathcal{L}_{rr} + \mathcal{L}_{rd} \tag{5.17}$$

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}.

5.2.1.6 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 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 5.3 and 5.6. We then follow

Devlin et al. (2019) 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 5.7 as follows.

$$S_{rd}(q_k^{rd}, p_j, s) = \max_{[\texttt{m}_\texttt{s}],[\texttt{m}_\texttt{e}] \in (q_k^{rd},\ p_j)} S_s(q_k^{rd}, p_j, [\texttt{m}_\texttt{s}]) \ + S_e(q_k^{rd}, p_j, [\texttt{m}_\texttt{e}]) \tag{5.18}$$

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 (Devlin et al., 2019). 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 5.1:

$$S(q_k, \{q_i\}_{i=1}^{k-1}, w, p_j, s) = S_{rt}(q_k^{rt}, p_j) \ + S_{rr}(q_k^{rd}, p_j) \ + S_{rd}(q_k^{rd}, p_j, s) \tag{5.19}$$

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

5.2.2 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 (Choi et al., 2019) that offers information-seeking conversations, (2) the CANARD dataset (Elgohary et al., 2019) 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 5.3. We will describe the data construction process in the following sections.

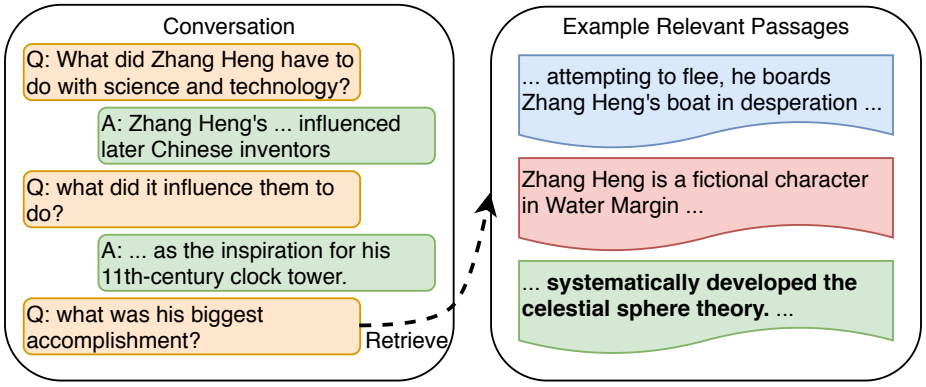


Figure 5.3: A partial OR-QuAC dialog and example passages retrieved from the collection by TF-IDF.

5.2.2.1 Self-contained Information-seeking Dialogs

We provided a description of QuAC in Section 4.3.1. 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.

Table 5.2: 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

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 (Elgohary et al., 2019). 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 5.2.

5.2.2.2 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.[5] The Wikipedia passages in QuAC were downloaded via PetScan[6] (Choi et al., 2019), 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[7] 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. (2019b). The split

[5]https://dumps.wikimedia.org/enwiki/20191020/

[6]http://petscan.wmflabs.org/

[7]https://github.com/attardi/wikiextractor

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 to be unanswerable during training. We do the split to make the passages fit for Transformer based retrievers and readers. Moreover, Yang et al. (2019b) 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.

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. However, in real-world scenarios, questions and answers in a dialog may be distributed at different locations of the corpus. This is a limitation of our dataset.

5.2.3 Experimental Setups

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

5.2.3.1 Competing Methods

To the best of our knowledge, there is no published work tackling the ORConvQA problem that we describe in Section 5.2.1.1. There is, however, a rich body of work

on single-turn open-domain QA, led by DrQA (Chen et al., 2017). 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 (Yang et al., 2019b). To be specific, the competing methods are:

- **DrQA** (Chen et al., 2017). 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.[8]

- **BERTserini** (Yang et al., 2019b). This model uses a BM25 retriever from Anserini[9] 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 work.

- **ORConvQA without history** (Ours w/o hist.). This is our model described in Section 5.2.1 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

[8]https://github.com/facebookresearch/DrQA

[9]http://anserini.io/

described in Section 5.2.1.3. This model is our adaptation of the open-retrieval QA framework (Lee et al., 2019b) 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 5.2.1.

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 settings.

5.2.3.2 Evaluation Metrics

In addition to the F1 and HEQ metrics described in Section 4.3.2.2, we also use 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 the reranker. MRR is a reflection of how well these two components contribute to the overall score in Equation 5.19. 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.

5.2.3.3 Implementation Details

Our models are implemented with PyTorch[10] and the open-source implementation of ALBERT and BERT by HuggingFace.[11]

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.

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 5.2.4.3. 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.

---

[10] `https://pytorch.org/`

[11] `https://github.com/huggingface/transformers`

Table 5.3: 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**‡

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

5.2.4 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.

5.2.4.1 Main Evaluation Results

We report the main evaluation results in Table 5.3. 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. (2019b) 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 Section 5.2.4.3. In addition, we observe that the reranker consistently outperforms the retriever. This suggests that although reranking is more expensive as it jointly models the

Table 5.4: 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‡

question and the passage, it enjoys better performance than the retriever that models the question and the passage separately.

5.2.4.2 Ablation Studies

Section 5.2.4.1 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 5.3 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 (Lee et al., 2019b; Das et al., 2019). In this ablation setting, we remove the reranking loss in Equation 5.17 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 5.19.

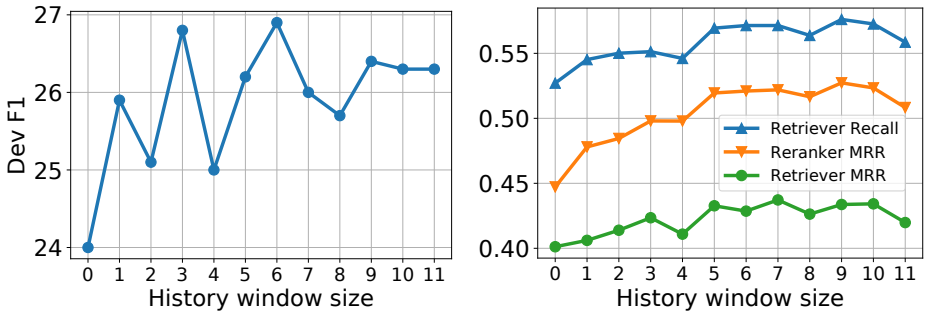
- **ORConvQA w/o learned retriever**. We replace our learned retriever with DrQA's TF-IDF retriever.
- **ORConvQA w/o first question (q) for retriever**. We do not manually include the first question of a dialog in the reformatted question for the retriever.

The ablation results are presented in Table 5.4. 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 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 5.2.4.1 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.



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

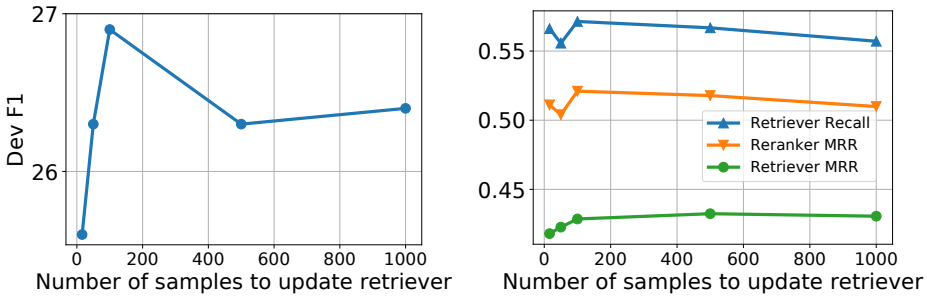
Figure 5.4: Impact of history window size w.

5.2.4.3 Additional Analyses

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 5.4.

In Figure 5.4a, 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 5.4b, 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 5.2.3.3, we show less degradation in the performance than previous work (Qu et al., 2019c) 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 (2018). 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



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

Figure 5.5: Impact of # samples to update retriever K_{rt}.

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 5.4a, suggesting that the fluctuation in the overall performance can be explained by the variation in retriever performance.

Impact of the number of passages to update retriever. Lee et al. (2019b) 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 5.2.3.3, 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 5.5.

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.

5.2.5 Discussions on Other Efforts

In this section, we describe two other efforts we made in attempt to further improve the performance of an ORConvQA system. Although these attempts do not yield better performance than our approach described in Section 5.2.1, they can inform future explorations in this direction.

5.2.5.1 End-to-End Joint Learning

As described in Section 5.2.1.5, our ORConvQA model employs a two-phase training approach: a retriever pretraining phase and a concurrent learning phase for the retriever and the reader. In the concurrent learning phase, although the retriever and the reader are updated simultaneously, the gradients from the reader loss does not back-propagate to the retriever. We investigate whether it is helpful to incorporate the supervision signals from the reader objective into the retriever training process in the concurrent learning phase.

For this purpose, we modify the reader loss to involve the retriever logit, i.e., $v_{\texttt{[m]}}$ in Equation 5.7 now becomes $v_{\texttt{[m]}} \oplus S_{rt}$, where $\oplus$ is the concatenation operation and S_{rt} is the retriever score computed with Equation 5.3. In this case, the reader will consider the relevance of a passage when predicting whether a token in this specific passage is the start/end token of the answer span. This aligns to our task better since the retriever can now be influenced by the overall QA performance. In other words,

the retriever and the reader are learned jointly in an end-to-end manner, rather than only learned concurrently.

Our experiments show the F1 scores to be 27.1 and 28.5 on the development set and the test set respectively, which are on par with our original results reported in Section 4.3.3. This suggests that it does not bring additional gains when we incorporate the supervision signals from the reader objective to the retriever training process. There are two potential reasons. First, it could be already sufficient to train the retriever with retriever pretraining and concurrent learning. The retriever does not benefit from the additional supervision signals from the reader objective. Second, simply concatenating the retriever score to the token representation can be less effective for the reader to consider the passage relevance. We will leave more effective approaches for end-to-end joint learning in our future work.

5.2.5.2 Post-domain Pretrained BERT

Post-domain pretraining, or domain-adaptive pretraining, refers to the approach to continue pretraining a language model on domain-specific data after it has been pretrained on the original LM pretraining domain (Gururangan et al., 2020). We investigate whether it is helpful to conduct post-domain pretraining for BERT on dialog data for our ORConvQA system.

We construct the post-domain pretraining data by combining dialogs from the QuAC (Choi et al., 2019) training set, the CoQA (Reddy et al., 2019) training set, and MSDialog-Complete (Section 3.2). This results in 54,302 dialogs with 721,830 utterances. We load the original BERT checkpoint (bert-base-uncased) and conduct further pretraining with our dialog data for 2 epochs using the masked language model loss (Devlin et al., 2019). We set the maximum sequence length to 512, the learning rate to 1e-5, the batch size to 4, and the number of warm up steps to 4,000. The other hyper-parameter settings follow the original BERT paper (Devlin et al., 2019).

Finally, we plug the post-domain pretrained BERT into the reader component of our ORConvQA system described in Section 5.2.1. The F1 scores are 26.1 and 27.3 on the development set and test set respectively, which are lower than our original results reported in Section 5.2.4. We speculate that the inferior results of the post-domain pretrained BERT can be explained by the deficiency of dialog data. Moreover, it also requires more investigation into the best approach to conduct post-domain pretraining when we want to use the model to jointly encode heterogeneous content (e.g., a conversation and a passage). We will leave this to future work.

5.3 Weakly-supervised Open-retrieval Conversational QA

5.3.1 Our Approach

In this section, we first formally define the task of open-retrieval ConvQA under a weak supervision setting. We then explain how we train the ORConvQA system described in Section 5.2 with our learned weak supervision approach.

5.3.1.1 Task Definition

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. Different from the task definition in full weak supervision in Section 5.2, we now assume no access to gold passages when training the reader. The gold passage for q_k is the passage in C that is known to contain or support a_k.

5.3.1.2 Weakly-Supervised Training

Overview. We follow the same architecture of the ORConvQA model in Section 5.2.[12] This section differs from Section 5.2 in how we train the model. Section 5.2

[12]We disable the reranker in Section 5.2 since our preliminary experiments indicated the weak supervision signals seem to lead to degradation for the reranker and the retriever.

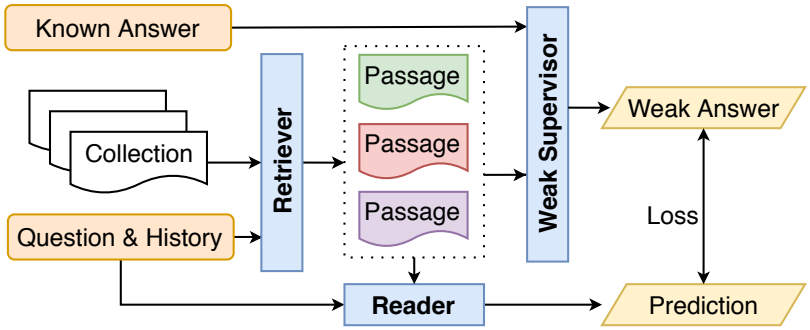


Figure 5.6: Overview of weak supervision for ORConvQA. Given a question and its conversation history, the retriever first retrieves top-K passages from the collection. The reader then reads the top passages and produces an answer. We adopt a weakly-supervised training approach. Given the known answer and one of the retrieved passages, the weak supervisor predicts a span in this passage as the weak answer to provide weak supervision signals for training the reader.

uses full supervision while this section adopts weak supervision. As described in Section 5.2, the training contains two phases, a pretraining phase for the retriever, and a concurrent learning phase for the reader and fine-tuning the question encoder in the retriever. Our weakly-supervised training approach is applied to the concurrent learning phase.

Figure 5.6 illustrates the role the weak supervisor plays in the system. Given a known answer a_k and one of the retrieved passages p_j, the weak supervisor predicts a span in p_j as the *weak answer* a_k^{weak}. This weak answer is the weak supervision signal for training the reader. The weak supervisor can also indicate there is no weak answer contained in p_j. A question is skipped if there are no weak answers in any of the retrieved passages.

Inspirations. Our learned weak supervision method is inspired by the classic span-match weak supervision. This method has been the default and only weak supervision method in previous open-domain QA research (Lee et al., 2019b; Chen et al., 2017; Das et al., 2019). These works mainly focus on factoid QA, where answers are short. A span-match weak supervisor can provide accurate supervision signals

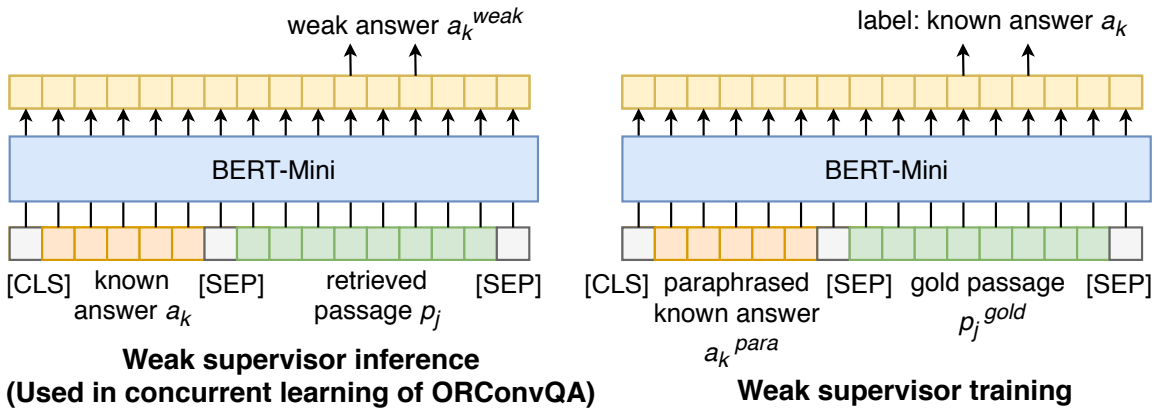


Figure 5.7: Learned weak supervisor. During the concurrent learning phase of ORConvQA, the weak supervisor conducts inference on a retrieved passage p_j (the left figure) to predict a passage span that is a paraphrase of the known answer a_k. When training of the weak supervisor (the right figure), the model is trained to predict the known answer a_k in the passage given a paraphrase of the known answer a_k^{para} and the passage.

since the weak answers are exactly the same as the known answers. In addition, the short answers can find matches easily in passages other than the gold ones. In information-seeking conversations, however, the answers can be long and freeform, and thus are more difficult to get an exact match in retrieved passages. Although the span-match weak supervisor can still provide accurate supervision signals in this scenario, it renders many training examples useless due to the failure to find exact matches. A straightforward solution is to find a span in a retrieved passage that has the maximum overlap with the known answer. Such overlap can be measured by word-level F1. This overlap method, however, can be intractable and inefficient since it has to enumerate all spans in the passage. This method also requires careful tuning for the threshold to output "no answer". Therefore, we introduce a learned weak supervisor based on Transformers (Vaswani et al., 2017) to predict a weak answer span directly in a retrieved passage given the known answer. This supervisor also has the ability to indicate whether a retrieved passage has a good weak answer or not.

Learned Weak Supervisor. Given the known answer a_k and one of the retrieved passages p_j, the weak supervisor predicts a span in p_j as the weak answer a_k^{weak}. Intuitively, a_k^{weak} is a paraphrase of a_k. We use a standard BERT-based extractive MC model (Devlin et al., 2019) here as shown in Figure 5.7, except that we use a_k for the question segment. The best weak answer for all top passages is the one with the largest sum of start and end token scores.

Although theoretically simple, this model presents challenges in training because position labels of a_k^{weak} are not available. Therefore, we consider the known answer a_k as the weak answer we are seeking since we know the exact position of a_k in its gold passage p_j^{gold}. We then use a diverse paraphrase generation model (described in the next section) to generate a paraphrase a_k^{para} for the known answer a_k. The paraphrase a_k^{para} simulates the known answer during the training of the weak supervisor, as shown in Figure 5.7. The weak supervisor is trained before concurrent learning and kept frozen during concurrent learning. We train the weak supervisor to tell if the passage does not contain a weak answer by pairing a randomly sampled negative passage with the known answer.

We are aware of a dataset, CoQA (Reddy et al., 2019), that provides both span answer and freeform answer for a given question q_k. In this case, we can take the freeform answer as a natural paraphrase a_k^{para} for the span answer (known answer) a_k when training the weak supervisor. For datasets that do not offer both answer types, our diverse paraphraser assumes the role of the oracle to generate the paraphrase answer. In other words, the use of the diverse paraphraser ensures that our weak supervision approach can be applied to a wide variety of conversation data that are beyond datasets like CoQA.

Diverse Paraphrase Model. We now briefly describe the diverse paraphraser (Krishna et al., 2020) used in the training process of the learned weak supervisor. This model is built by fine-tuning GPT2-large (Radford et al., 2019) using encoder-free

seq2seq modeling (Wolf et al., 2018). As training data we use PARANMT-50M (Wieting and Gimpel, 2018), a massive corpus of back translated data (Wieting and Gimpel, 2018). The training corpus is aggressively filtered to leave sentence pairs with high lexical and syntactic diversity so that the model can generate diverse paraphrases. We refer our readers to Krishna et al. (Krishna et al., 2020) for further details.

5.3.2 Experiments

We now describe the experimental setup and report the results of our evaluations.

5.3.2.1 Experimental Setup

Dataset. We select two ConvQA datasets, QuAC (Choi et al., 2019) and CoQA (Reddy et al., 2019), with different answer types (span/freeform) to conduct a comprehensive evaluation of our weak supervision approach and to provide insights for weakly-supervised ORConvQA. We present the data statistics of both datasets in Table 5.5. We remove unanswerable questions in both datasets since there is no basis to find weak answers.[13] Specifically, the two datasets are: (1) **OR-QuAC (span answers)**. We use the OR-QuAC dataset introduced in Section 5.2.2. This dataset adapts QuAC to an open-retrieval setting. It contains information-seeking conversations from QuAC, and a collection of 11 million Wikipedia passages (document chunks). (2) **OR-CoQA (freeform answers)**. We process the CoQA dataset (Reddy et al., 2019) in the Wikipedia domain for the open-retrieval setting in a similar manner with OR-QuAC, resulting in the OR-CoQA dataset. CoQA offers freeform answers generated by people in addition to span answers, resulting in more natural conversations. OR-CoQA and OR-QuAC share the same passage collection. Similar to QuAC, many initial questions in CoQA are also ambiguous and hard to in-

[13]This difference in the data accounts for the discrepancies of the full-supervision results presented in Table 5.6.

Table 5.5: Data statistics of OR-CoQA and OR-QuAC (weak supervision).

Items	OR-CoQA			OR-QuAC		
	Train	Dev	Test	Train	Dev	Test
# Dialogs	1,521	100	100	4,383	490	771
# Questions	23,027	1,494	1,611	25,824	2,808	4,406
# Avg. Question Tokens	5.8	5.7	5.8	6.8	6.6	6.8
# Avg. Answer Tokens	2.8	2.6	2.6	15.0	15.0	14.7
# Avg. Dialog Questions	15.1	14.9	16.1	5.9	5.7	5.7
# Avg. Max History Turns per Question	7.9/22	7.6/21	7.9/19	2.8/11	2.8/11	2.8/11

terpret without the given gold passage (e.g., "When was the University established?"). In Section 5.2.2, we deal with this by replacing the *first question* of a conversation with its context-independent *rewrite* offered by the CANARD dataset (Elgohary et al., 2019) (e.g., "When was the University *of Chicago* established?"). This makes the conversations self-contained. Since we are not aware of any CANARD-like resources for CoQA, we prepend the document title to the first question for the same purpose (e.g., "*University of Chicago* When was the University established?"). Since the CoQA test set is not publicly available, we take the original development set as our test set and 100 dialogs from the original training set as our development set.

Competing Methods. Since this work focuses on weak supervision, we use the same ORConvQA model and vary the supervision methods. To be specific, the competing methods are:

- **Full supervision** (Full S): Manually add the gold passage to the retrieval results and use the ground-truth answer span (Section 5.2). This only applies to QuAC since we have no passage relevance for CoQA. This method is not comparable with other weak supervision methods that do not have access to the groundtruth answers in concurrent learning.

- **Span-match weak supervision** (Span-match WS): This method finds a weak answer span that is identical to the known answer in the retrieved passages. When there are multiple matched spans, we take the first one.

- **Learned weak supervision** (Learned WS): This is our method in Section 5.3.1.2 that finds a paraphrased span of the known answer as the weak answer.

- **Combined weak supervision** (Combined WS): This is the combination of the above two methods. We first use the span-match weak supervisor to try to find a weak answer. If it fails, we take the weak answer found by the learned weak supervisor.

Evaluation Metrics. We use the same ConvQA metrics (F1 and HEQ) described in Section 4.3.2.2. In addition, we define another set of metrics to reveal the impact of the weak supervisor in the training process as follows. **% Has Answer** is the percentage of training examples that have a weak answer (in the last epoch). **% Hit Gold** is the percentage of training examples that have a weak answer identified in gold passages (in the last epoch). **Recall** is the percentage of training examples that have the gold passage retrieved (in the last epoch). **% From Gold** is the percentage of predicted answers that are extracted from the gold passages.

Implementation Details. Our models are based on the open-source implementation of the Diverse Paraphrase Model[14], and the HuggingFace Transformers repository.[15] We use the same pretrained retriever in Section 5.2 for both datasets. For concurrent learning of ORConvQA, we set the number of training epochs to 5 (larger than Section 5.2) to account for the skipped steps where no weak answers are found. We set the number of passages to update the retriever to 100, and the history window size to 6 since these are the best settings reported in Section 5.2. The max

[14] https://github.com/martiansideofthemoon/style-transfer-paraphrase

[15] https://github.com/huggingface/transformers

Table 5.6: Evaluation results on OR-QuAC (span answers) with weak supervision. The learned weak supervisor causes no statistical significant performance decrease compared span match.

Methods		Full S	Span-match WS	Learned WS	Combined WS
Train	% Has Answer	100.00%	72.96%	75.98%	75.52%
Dev	F1	**22.8**	**20.8**	20.2	20.1
	HEQ-Q	**8.1**	**6.8**	6.0	6.4
	HEQ-D	0.6	0.6	0.2	0.6
Test	F1	**23.9**	**23.6**	23.1	23.2
	HEQ-Q	**14.0**	12.3	11.8	**12.5**
	HEQ-D	**2.2**	1.7	**1.9**	**1.9**

answer length is set to 40 for QuAC and 8 for CoQA. The rest of the hyper-parameters and implementation details for the ORConvQA model are the same as in Section 5.2.

For the weak supervisor, we use BERT-Mini (Turc et al., 2019) for better efficiency. We set the number of training epochs to 4, the learning rate to 1e-4, and the batch size to 16. As discussed in Section 5.3.1.2, the diverse paraphraser is used for OR-QuAC only. For OR-CoQA, we use the freeform answer provided by the dataset as a natural paraphrase to the span answer.

5.3.2.2 Evaluation Results on Span Answers

Given the different properties of span answers and freeform answers, we study the performance of our weak supervision approach on these answers separately. We report the evaluation results on the span answers in Table 5.6. Our observations can be summarized as follows.

The full supervision setting yields the best performance, as expected. This verifies the supervision signals provided by the gold passages and the ground-truth answer spans are more accurate than the weak ones. All supervision approaches have similar performance on span answers. This suggests that span-match weak supervision is sufficient to handle conversations with span answers. Ideally, if the known answer

is part of the given passage, the learned weak supervisor should be able to predict the weak answer as exactly the same with the known answer. In other words, the learned weak supervisor should fall back to the span-match weak supervisor when handling span answers. In practice, this is not guaranteed due to the variance of neural models. However, our learned weak supervisor causes no statistical significant performance decrease compared with the span-match supervisor. This demonstrates that the learned weak supervision approach can cover span answers as well. Although we observe that the learned supervisor can identify more weak answers than span match, these weak answers could be false positives that do not contribute to the model performance. Finally, for the combined weak supervisor, our analysis shows that 96% of the weak answers are identified by span match, further explaining the fact that all weak supervision approaches have almost identical performance.

5.3.2.3 Evaluation Results on Freeform Answers

We then look at the evaluation results on freeform answers in Table 5.7. These are the cases where a span-match weak supervisor could fail. We observe that combining the learned weak supervisor with span match brings a statistically significant improvement over the span-match baseline on the test set, indicating these two methods complement each other. The test set has multiple reference answers per question, making the evaluation more practical. In addition, the learned supervisor can identify more weak answers than span match, these weak answers contribute to the better performance of our model. Further, for the combined weak supervisor, our analysis shows that 77% of the weak answers are identified by span match. This means that nearly a quarter of the weak answers are provided by the learned supervisor and used to improve the performance upon span match. This further validates the source of effectiveness of our model.

Table 5.7: Evaluation results on OR-CoQA (freeform answers) with weak supervision. ‡ means statistically significant improvement over the span-match baseline with $p < 0.05$.

Methods		Span-match WS	Learned WS	Combined WS
Train	% Has answer	51.81%	65.75%	70.35%
Dev	F1	18.3	18.9	**19.7**
	HEQ-Q	11.6	9.0	**12.7**
	HEQ-D	0.0	0.0	0.0
Test	F1	24.3	26.0	**28.8**‡
	HEQ-Q	19.9	15.9	**22.5**
	HEQ-D	0.0	0.0	0.0

Table 5.8: A closer look at the weakly-supervised training process for OR-QuAC.

Methods	Train			Dev	Test
	% Has Ans	% Hit Gold	Recall	% From Gold	% From Gold
Full S	100.00%	100.00%	1.0000	45.23%	27.46%
Span-match WS	72.96%	68.97%	0.7190	40.88%	28.80%
Learned WS	75.98%	67.24%	0.7187	39.89%	28.73%
Combined WS	75.52%	68.37%	0.7129	40.28%	28.39%

5.3.2.4 A Closer Look at the Training Process

We take a closer look at the training process, as shown in Table 5.8. We conduct this analysis on OR-QuAC only since we do not have the ground-truth passage relevance for CoQA. We observe that, "% Has Ans" are higher than "% Hit Gold" for all weak supervision methods, indicating all of them can identify weak answers in passages beyond the gold passages. In particular, our method can identify more weak answers than span match. We also notice that "% Hit Gold" is only slightly lower than "Recall", suggesting that most of the retrieved gold passages can yield a weak answer. This verifies the capability of the weak supervisors. Finally, "% From Gold" are relatively low for all methods, indicating great potential for improvements.

Table 5.9: Case study for learned weak supervision. Boldface denotes discrepancies and italic denotes paraphrasing.

	#		Questions and Answers
Good	1	Question	Where was the album released?
		Known answer	on online forums and music sites.
		Weak answer	on online forums and music sites.
	2	Question	... mention anything else he starred in?
		Known answer	After starring ... the film adaptation of The Music Man
		Weak answer	After starring ... film adaptation of The Music Man **(1962)**.
	3	Question	Where did he distribute the Cocaine?
		Known answer	flying out planes several times, mainly between Colombia and Panama, along smuggling routes into the United States.
		Weak answer	*He flew a plane himself several times, mainly between Colombia and Panama, in order to smuggle a load into the United States.*
Bad	4	Question	how long have people had clothes?
		Known answer	as long ago as 650 thousand years ago
		Weak answer	around **170,000** years ago.
	5	Question	What is data compression called?
		Known answer	reducing the size of a data file
		Weak answer	**By using wavelets, a compression ratio**

5.3.2.5 Case Study and Error Analysis

We then conduct a qualitative analysis by presenting weak answers identified by the learned weak supervisor in Table 5.9 to better understand the weak supervision process. Example 1 and 2 show that our learned weak supervisor can find weak answers that are exactly the same or almost identical to the known answers when an exact match of the known answer exits, further validating our method can potentially cover span-match weak supervision. Example 3 shows that if an exact match does not exist, our method can find a weak answer that expresses the same meaning with the known answer. This is a case that a span-match weak supervisor would fail.

Example 4 shows that our method tends to focus on the lexical similarity only but gets the fact wrong. Example 5 indicates our method sometimes finds a weak answer that is relevant to the known answer but cannot be considered as a good answer. These are the limitations of our method.

5.4 Summary

In this chapter, 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. We study both full supervision and weak supervision approaches for training.

In the fully-supervised setting, 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.

In the weakly-supervised setting, we propose a learned weak supervision approach for open-retrieval conversational question answering. Our experiments on two datasets show that, although span-match weak supervision can handle span answers, it is not sufficient for freeform answers. Our learned weak supervisor is more flexible since it can handle both span answers and freeform answers. It is more powerful when combined with the span-match supervisor.

5.5 Towards Real-world ORConvQA

Finally, we briefly discuss a basket of engineering enhancements we would like to make if we want to adapt our ORConvQA system to real-world scenarios.

- Encoder. We build our model on top of BERT, one of the most widely-used pretrained language models, to test our hypothesis. For practical applications, one might want to compare the performance of different pretrained language

models that emphasize robustness (Liu et al., 2019b), efficiency, and the ability to model long documents (Yang et al., 2020b) (the latter two are often tackled together with efficient transformer architectures (Wang et al., 2020; Kitaev et al., 2020; Zaheer et al., 2020; Beltagy et al., 2020)). Efficiency, especially that at inference/serving time, is vital to the productization of an ORConvQA system since the customers prefer responses in real time. The ability to model long documents, which may also employ techniques for efficiency, is also desired for QA tasks because longer context (richer knowledge) is beneficial to the QA performance.

- Hybrid retrieval. Sparse retrieval emphasizes term match, which could be beneficial when the user is trying to refind (Dumais et al., 2003) a passage or an answer. Dense retrieval, on the other hand, focuses on semantic match, which is especially suitable for information-seeking tasks, where the information need can be highly under-defined and exploratory. Combining the power of sparse retrieval and dense retrieval for a hybrid retrieval approach (Luan et al., 2020) can potentially lead to performance improvement. Within the scope of dense retrieval, one might want to investigate the performance of more advanced late interaction approaches (Khattab and Zaharia, 2020; Chen et al., 2020) beyond dot product.

- Training. Models trained with diverse and large-scale training data often have better generalization abilities. Since the gold training labels can be expensive to acquire, one might want to first train with large-scale data in a unsupervised or weakly-supervised manner, followed by training with a small amount of human-annotated data. In our circumstances, we can train the weak supervisor introduced in Section 5.3.1 by firstly using the training data generated with a high-quality diverse paraphraser, followed by training with a handful of

human-generated data. Similarly, the concurrent learning phase described in Section 5.2.1 can also first employ weak supervision (Section 5.3) and then full supervision (Section 5.2).

- Answer generation. The ability to generate natural and fluent answers is also desirable in an ORConvQA system. This ability is indispensable when we want to aggregate multiple answer spans (potentially from different passages and documents) (Liu and Lapata, 2019) or to summarize tabular data (Zhang et al., 2020).

- Wider applicability. To make the ORConvQA system apply to a larger audience and wider applications, the system should have the ability to take in information needs that are multilingual (Khanuja et al., 2020; Sorg and Cimiano, 2012), cross-lingual (Jiang et al., 2020; Lignos et al., 2019; Sorg and Cimiano, 2012), and even beyond the text form (Marino et al., 2019; Lien et al., 2020) (e.g., an input may contain both texts and images). These can be done by swapping the encoders in an ORConvQA system with multilingual (Devlin et al., 2019) or multi-modal encoders (Tan and Bansal, 2019).

CHAPTER 6

HISTORY MODELING FOR CONVERSATIONAL RE-RANKING

6.1 Introduction

Previous chapters approach conversational IR via the proxy of human conversations. Although they all model conversation history, the previous chapters do not address the issue of how to incorporate user behavior. In this chapter, we investigate the interactive and iterative property of conversational search via the session search task, where we focus on incorporating history user behavior into a document ranking system.

To fulfill a complicated information need with a search engine, users typically need to conduct searches for multiple turns. Temporally connected turns are referred to as a *session* or *task* (Jones and Klinkner, 2008; Wang et al., 2013). In each turn, the user issues or reformulates a query, browses search engine result pages (SERPs), and clicks on one or more documents for further investigation. This iterative information seeking process bears a strong resemblance to conversational search. The rich user behavior and interaction data for document ranking is readily available from search logs in commercial search engines on a large scale. Moreover, evaluation methods for document ranking are much more mature than that of conversational search. Therefore, we argue that the conversational document ranking task should be one of the core steps towards building functional conversational search systems.

The idea of modeling history or context for document ranking has been studied in many works (White et al., 2013, 2010; Xiang et al., 2010; Hagen et al., 2013; Cui

and Cheng, 2014; Technology et al., 2013; Bennett et al., 2012) and was particularly featured in the Session Track (Carterette et al., 2016) in TREC (Text REtrieval Conference). Due to the lack of large-scale session datasets, most approaches are mainly limited to non-parametric or feature-based models (Hagen et al., 2013; Cui and Cheng, 2014; Technology et al., 2013; Shen et al., 2005; White et al., 2013, 2010; Xiang et al., 2010). These models may not be able to capture the complex user intent dynamics in a real search session. Recently, many large-scale datasets (Lowe et al., 2015; Zheng et al., 2018) have greatly boosted the research progress for *neural* IR, resulting in a wide range of model architectures (Guo et al., 2016; Hu et al., 2014; Huang et al., 2013; Shen et al., 2014; Yang et al., 2016; Mitra et al., 2017; Xiong et al., 2017; Dai et al., 2018; Yang et al., 2018; Qu et al., 2019a) for matching and retrieval. Consequently, IR researchers have begun to revisit the issue of conversational document ranking with neural models (Ahmad et al., 2018, 2019b). In this chapter, we introduce behavior awareness to a neural ranker. In a real search scenario, a user's past behaviors may contribute differently when determining the relevance of the current document. For example, behaviors in the immediate previous turn could be more informative than those in a distant turn. Moreover, a previous clicked document and a previous skipped document could provide different clues of the information need. It is essential for a neural ranker to be aware of and distinguish different user behaviors in the session. For this purpose, we design the Hierarchical Behavior Aware Transformers (HBA-Transformers) that feature a *hierarchical behavior attention mechanism* to enable flexible incorporation of the session history. Our system is built on top of the state-of-the-art BERT model to leverage its power in document ranking. Given a history of relevance feedback behaviors, we first use BERT to generate contextualized representations for the input sequence on a token level. Then an *intra-behavior attention* layer aggregates tokens from same behaviors to generate behavior-level representations. These representations are then enhanced

by the *behavior aware embeddings*, which identify the type and position properties of a user behavior for effective characterization and modeling. We then apply the *inter-behavior attention* layer to enable bidirectional session modeling. This layer attends to all behavior representations in the session along with the current document and produces a query-document representation augmented by session information. This is followed by a document ranker to predict a relevance score.

We conduct extensive experiments on the AOL session search dataset constructed in previous research (Ahmad et al., 2019b). We first show that a BERT based ranker without any context information is able to outperform a recent context aware recurrent model (Ahmad et al., 2019b). This improvement demonstrates the capability of BERT in single-turn ranking. We further show that BERT is capable of modeling session history by simply prepending history user behaviors to the current query. This method is conceptually simple yet highly effective.

Moreover, we further demonstrate that our hierarchical behavior attention mechanism is significantly more powerful in this scenario than a simple concatenation. This indicates that behavior awareness plays an important role in conversational document ranking. Finally, we conduct an in-depth analysis on the conversational property of queries with the publicly available Microsoft Generic Intent Encoder API (Zhang et al., 2019).[1] We show that coherent sessions tend to be more *conversational* as they are more demanding in considering the user behaviors in the session history. Our HBA-Transformers can be potentially triggered at runtime upon identifying coherent sessions.

[1] https://msturing.org/

6.2 Behavior Aware Transformers

In this section, we first set up the task of conversational re-ranking following conventions. We then describe our Hierarchical Behavior Aware Transformers and explain the intuitions behind our model.

6.2.1 Task Definition

The conversational re-ranking task is defined as follows. We are given search logs that are organized in sessions. A search session is denoted as $\{T^i\}_{i=1}^N$, where T^i is the i-th turn in the session and N is the total number of turns. T^i is further denoted as a triple $\{q^i, D^i, Y^i\}$, where q^i is the search query in this session, and $D^i = \{d_j^i\}$ is the retrieved top-k documents for $q^i (1 \leq j \leq k)$. D^i is typically ranked by an initial retrieval method and j denotes the rank. $Y^i = \{y_j^i\}$ is a set of binary relevance labels (*i.e.* $y_j^i \in \{0, 1\}$) for D^i. The click label $y_j^i = 1$ means the user makes a "satisfied click" on d_j^i after issuing q^i. Given the current query q^n, the candidate documents D^n, and the search history $H^n = \{T^i\}_{i=1}^{n-1}$ up to the current turn, the task is to rerank D^n so that the clicked documents are ranked as high as possible.

In a history turn $T^i = \{q^i, D^i, Y^i\}$, the candidate documents D^i and its corresponding click labels Y^i reveal the *clicked* document d_+^i and the *skipped* document d_-^i for turn i. The skipped document d_-^i is identified by a commonly used strategy known as *Skip Above and Skip Next* (Agichtein et al., 2006). For example, if the second document is clicked, then the first and third documents are considered skipped. When there are multiple skipped documents, we take the first one. The skipped document d_-^i is less relevant to the information need. Therefore, we use a history turn T^i as three user behaviors: $\{q^i, d_+^i, d_-^i\}$.

Given all available history turns $H^n = \{T^i\}_{i=1}^{n-1}$, we extract a sequence of *history user behaviors* denoted as $H_*^n = \{q^1, d_+^1, d_-^1, q^2, d_+^2, d_-^2, \ldots, q^{n-1}, d_+^{n-1}, d_-^{n-1}\}$. We fur-

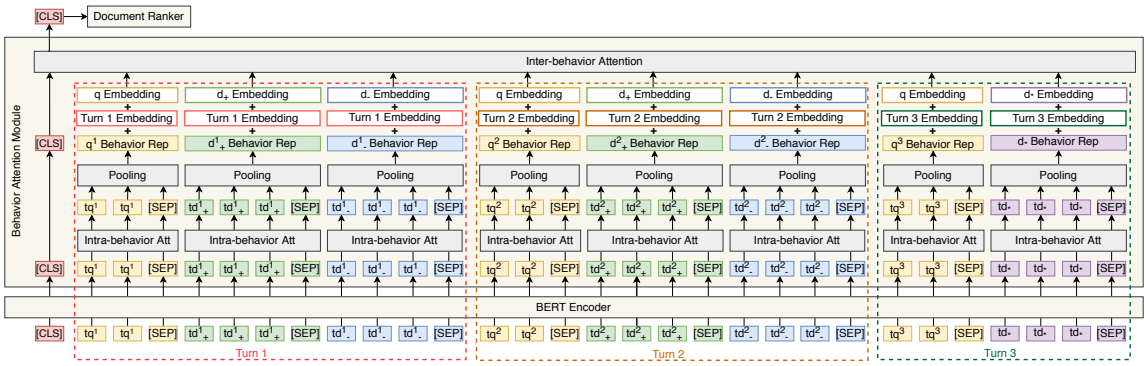


Figure 6.1: Model architecture. Suppose the current query is q^3, and the given document is d. The input sequence is the concatenation of session user behaviors and the given document. tq, td denote query/document tokens. The intra-behavior attention layer generates behavior representations from token representations produced by BERT. Then the inter-behavior attention layer attends to all behaviors with a holistic view of the session. Finally, the ranking module predicts the relevance score.

ther define a set of *session user behaviors* as $H^n_* \cup \{q^n\}$, to include the current query q^n. The task is to rerank D^n given $H^n_* \cup \{q^n\}$.

6.2.2 Model

We present the Hierarchical Behavior Aware Transformers (HBA-Transformers) in this section. As shown in Figure 6.1, our model consists of a *BERT encoder*, a *hierarchical behavior attention module*, and a *document ranker*. The BERT encoder encodes input tokens into contextualized representations. Then the hierarchical behavior attention module first generates behavior representations and then attends to these representations with a holistic view of the session. Finally, the document ranking module predicts a relevance score of the candidate document. This hierarchical architecture enables flexible integration of the session history and thus can be more effective in conversational re-ranking. We illustrate each module in detail in the following sections.

6.2.2.1 BERT Encoder

As shown in the lower part of Figure 6.1, the encoder is a BERT model that encodes contextual information for session user behaviors and the current candidate document. BERT is a pre-trained language model that is designed to learn deep contextual representations using Transformers (Vaswani et al., 2017). It has shown state-of-the-art performance in document ranking (Dai and Callan, 2019; Nogueira and Cho, 2019).

Previous works (Dai and Callan, 2019; Nogueira and Cho, 2019; MacAvaney et al., 2019) apply BERT for ranking in a manner of sequence pair classification, where the input sequence is "`[CLS]` q `[SEP]` d `[SEP]`". `[CLS]` and `[SEP]` are special tokens introduced in BERT. We extend this scheme by prepending history user behaviors in a window size of w to the query segment, i.e., "`[CLS]` q^{n-w} `[SEP]` d_{+}^{n-w} `[SEP]` d_{-}^{n-w} `[SEP]` $\cdots$ `[SEP]` q^{n} `[SEP]` d `[SEP]`". Each segment can have multiple tokens. Let $\mathbf{t}_m \in \mathbb{R}^h$ denotes the embedding representation of the m-th token in the input sequence and h denotes the hidden dimension size. The input sequence is denoted as $\{\mathbf{t}_m\}_{m=1}^{M}$, where M denotes the sequence length. BERT outputs a hidden representation for every token:

$$\{\hat{\mathbf{t}}_m\}_{m=1}^{M} = \text{BERT}(\{\mathbf{t}_m\}_{m=1}^{M}) \tag{6.1}$$

The contextualized token representation $\hat{\mathbf{t}}_m$ is obtained by attending to every single token in the input sequence and thus is contextualized in terms of the entire available session and the candidate document. This is the advantage of modeling the concatenation of the session over modeling each behavior individually.

6.2.2.2 Hierarchical Behavior Attention Module

We design a behavior attention module to learn to attend to user behaviors and the current document to produce a history-enhanced query-document representation

for ranking. The hierarchy in the behavior attention module is composed of an intra-behavior attention layer and an inter-behavior attention layer.

Intra-behavior Attention. As shown in the middle part of Figure 6.1, the input of the intra-behavior attention layer is the token-level representations produced by BERT, and the output is behavior-level representations for all behaviors. In our architecture, the BERT encoder computes attention on a token level while the inter-behavior attention layer computes attention on a behavior level. The intra-behavior attention layer serves as a connecting component of these two modules.

Given a specific session user behavior (or the current document), we isolate the representations of tokens within this behavior as $\{\hat{\mathbf{t}}_m\}_{m=s}^{e}$, where s and e are the start and end of this behavior. The `[CLS]` of the input sequence and the trailing `[SEP]` of this behavior are also considered as within-behavior tokens. We then apply an intra-behavior attention layer that has the same structure with a BERT layer. Specifically, this has three sub-layers: an attention layer (Eq. 6.2), an intermediate layer (Eq. 6.3), and an output layer (Eq. 6.4):

$$\{\tilde{\mathbf{t}}_m^1\} = \text{LayerNorm}\ (\{\hat{\mathbf{t}}_m\}_{m=s}^{e} + \text{FFN}\ (\text{MHAtt}\ (\{\hat{\mathbf{t}}_m\}_{m=s}^{e}))) \tag{6.2}$$

$$\{\tilde{\mathbf{t}}_m^2\} = \text{GELU}\ (\text{FFN}\ (\{\tilde{\mathbf{t}}_m^1\})) \tag{6.3}$$

$$\{\tilde{\mathbf{t}}_m\} = \text{LayerNorm}\ (\{\tilde{\mathbf{t}}_m^2\} + \text{FFN}\ (\{\tilde{\mathbf{t}}_m^2\})) \tag{6.4}$$

where LayerNorm is the layer normalization (Ba et al., 2016), MHAtt is the multi-head attention mechanism (Vaswani et al., 2017), GELU is the Gaussian error linear units (Hendrycks and Gimpel, 2016), and FFN is a feed forward layer. $\{\tilde{\mathbf{t}}_m\}$ denotes intermediate representations for tokens in this behavior.

Finally, we aggregate the token representations using an average pooling on the dimension of sequence length:

$$\mathbf{r} = \text{AvgPooling}\ (\{\tilde{\mathbf{t}}_m\}) \tag{6.5}$$

where $\mathbf{r} \in \mathbb{R}^h$ denotes the representation for this session user behavior (or the current candidate document). The encoding of different behaviors are vectorized to produce a set of behavior representations $\mathbf{R} = \{\mathbf{r}_{q^{n-w}}, \mathbf{r}_{d_+^{n-w}}, \mathbf{r}_{d_-^{n-w}}, \cdots, \mathbf{r}_{q^n}, \mathbf{r}_d\}$ for all behaviors in this session and the given candidate document.

Inter-behavior Attention. As shown in the upper part of Figure 6.1, given the behavior representations $\mathbf{R}$ produced by the intra-behavior attention layer, the inter-behavior attention layer considers each individual behavior as a whole and compute their attention to each other. We further introduce *behavior awareness* to the inter-behavior attention layer by proposing the *behavior aware embeddings*. Since the behavior position and type are two properties that can uniquely identify a user behavior in a session, we design two sets of behavior aware embeddings, namely, the *behavior position embeddings* and the *behavior type embeddings*, to let the model be aware of these properties.

For the behavior position embeddings, the position of a user behavior refers to its relative position in terms of the current query. We use different embeddings for behaviors at different positions. The behavior type embeddings work in a similar manner. The vocabulary of behavior types is defined as $\{q, d_+, d_-, d_*\}$, where d_* denotes the current document, whose relevance has not been judged. The behavior aware embeddings are randomly initialized and learned. For each behavior, the behavior aware embeddings are added to the behavior representations followed by a layer normalization. For example, an enhanced behavior representation for $\mathbf{r}_{d_+^{n-w}}$ is computed as follows:

$$\hat{\mathbf{r}}_{d_+^{n-w}} = \text{LayerNorm}\left(\mathbf{r}_{d_+^{n-w}} + \mathbf{e}_{n-w} + \mathbf{e}_{d_+}\right) \tag{6.6}$$

where $\mathbf{e}_{n-w} \in \mathbb{R}^h$ is the behavior position embedding for turn $n-w$ and $\mathbf{e}_{d_+} \in \mathbb{R}^h$ is the behavior type embedding for d_+. The `[CLS]` representation is prepended to the session user behaviors and the current document so that we can follow the same

pooling strategy of Devlin et al. (2019). The behavior aware embeddings for [CLS] are set to the same ones as the current document.

We then apply the same transformations as in Eq. 6.2, 6.3, and 6.4 to obtain $\tilde{\mathbf{R}} = \{\tilde{\mathbf{r}}_{\texttt{[CLS]}}, \tilde{\mathbf{r}}_{q^{n-w}}, \tilde{\mathbf{r}}_{d_+^{n-w}}, \tilde{\mathbf{r}}_{d_-^{n-w}}, \cdots, \tilde{\mathbf{r}}_{q^n}, \tilde{\mathbf{r}}_d\}$. Different from the intra-behavior attention layer that computes attention on a token level, these transformations are now performed on a behavior level for bidirectional session modeling. Lastly we obtain a final representation $\mathbf{r}^*_{\texttt{[CLS]}}$ for ranking by taking a linear projection of the hidden states corresponding to the [CLS] token as follows:

$$\mathbf{r}^*_{\texttt{[CLS]}} = \text{FFN}\ (\tilde{\mathbf{r}}_{\texttt{[CLS]}}) \tag{6.7}$$

This representation is considered a history-enhanced query-document representation for ranking.

6.2.2.3 Document Ranker

The document ranking module is the last module in Figure 6.1. Following previous work (Nogueira and Cho, 2019), we use a linear layer to obtain the probability of d being relevant as follows:

$$\Pr\ (y \mid \mathbf{r}^*_{\texttt{[CLS]}}) = \text{softmax}\ (\text{FFN}\ (\mathbf{r}^*_{\texttt{[CLS]}})) \tag{6.8}$$

where $y \in \{0, 1\}$ is the relevance label. At training time, we use binary classification with cross-entropy loss. At testing time, we obtain the final list of documents for each query by ranking the documents with respect to the probability of relevance.

6.3 Experimental Settings

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

Table 6.1: Data Statistics. We follow previous works (Ahmad et al., 2019b; Gao et al., 2010; Huang et al., 2018b, 2013; Dai et al., 2018; Zheng et al., 2018) to use the document title as its content.

Data Split	Train	Valid	Test
# Sessions	219,748	34,090	29,369
# Queries	566,967	88,021	76,159
# Avg. Queries per Session	2.58	2.58	2.59
Avg. Query Length	2.86	2.85	2.9
Avg. Doc Length	7.27	7.29	7.08
# Avg. Doc per Query	5	5	50
# Avg. Click per Query	1.08	1.08	1.11
# Min/Med/Max Queries per Session	2/2/10	2/2/10	2/2/10

6.3.1 Dataset Description

We experiment with the AOL search log data created and used in Ahmad et al. (2019b) for easy and fair comparisons with their methods. This dataset is 200 times larger than the Session Track data and is thus much more suitable for training deep models. We present statistics of the dataset in Table 6.1.

The AOL query log (Pass et al., 2006) only contains queries and their clicked documents. It does not record other candidate documents returned to the users. Ahmad et al. (2019b) construct candidate documents for each query by sampling from the top documents retrieved by BM25 (Robertson and Zaragoza, 2009). Then they group user query logs into sessions by computing the cosine similarity of query representations. The query representations are obtained by averaging the query token embeddings from GloVe (Pennington et al., 2014). They use a cosine similarity threshold of 0.5 to segment the sessions and discard the search sessions with less than two queries. They use the document title as its content following previous studies (Gao et al., 2010; Huang et al., 2018b, 2013; Dai et al., 2018; Zheng et al., 2018). Although this is a limitation, previous work (Zamani et al., 2017) indicates that the title field is very informative in re-ranking.

6.3.2 Experimental Setup

6.3.2.1 Competing Methods

Ahmad et al. (2019b) have shown that their method significantly outperform both classical and neural ad-hoc retrieval models. Therefore, we first compare with their method, the best performing approach in this task. We then build another context aware neural model based on BERT as an even stronger baseline. To be specific, the competing methods are:

- **CARS** (Ahmad et al., 2019b): This model uses RNN (Recurrent Neural Network) based encoders to encode the queries and clicks in a session into hidden representations. It then uses two recurrent structures to summarize these representations. It is trained with a multi-task learning setting for both conversational document ranking and query suggestion. It is by far the strongest published baseline in this task and significantly outperforms traditional IR methods and previous neural models.

- **BERT** (Nogueira and Cho, 2019): Nogueira and Cho (2019) applies BERT to document ranking. This model only considers the current query and the candidate document. Specifically, they use BERT to model (q_n, d) with the standard setting of using "`[CLS]` q_n `[SEP]` d `[SEP]`" as the input sequence. This module is then followed by the same document ranker described in Section 6.2.2.3.

- **BERT-Concat**: This is the HBA-Transformers without the hierarchical behavior attention module. Given the token level representation produced by the BERT encoder, we take a linear projection of the `[CLS]` representation and apply the document ranker directly. In order to investigate the utilities of different behavior types, we design several variations of BERT-Concat that only consider specific user behaviors, namely, **-Q**, **-QC**, and **-QCS**. "Q", "C",

"S" denote previous **q**ueries, previous **c**licked documents, and previous **s**kipped documents respectively.

- **HBA-Transformers**: This refers to the model with the hierarchical behavior attention mechanism enhanced by the behavior aware embeddings introduced in Section 6.2. This also has variations of -QC and -QCS.

6.3.2.2 Evaluation Metrics

We follow Ahmad et al. (2019b) and use mean reciprocal rank (MRR@all) and normalized discounted cumulative gain (nDCG@1, 3, 10) for evaluation. Normalized session discounted cumulative gain (nsDCG) (Järvelin et al., 2008) and session average precision (sAP) (Kanoulas et al., 2011), metrics for session search, assume that a session is a series of query reformulations. In AOL data, queries in a session are not necessarily related. So we mainly focus on MRR and nDCG.

6.3.2.3 Implementation Details

Our models are implemented with PyTorch[2] and based on the open-source implementation of BERT by Hugging Face.[3] We use the BERT-Base (uncased) model. We use the same training scheme for all our models, including BERT/BERT-Concat and HBA-Transformers. We set the max sequence length to 128, the training batch size to 512, the number of training epochs to 10, and the learning rate to 1e-4. The warm up portion of the learning rate is 10% of the total steps. We set the gradient accumulation steps for BERT/BERT-Concat and HBA-Transformers to 1 and 4 respectively. The history window size is set to 3. We use half precision for training as suggested in the Hugging Face repository to alleviate CUDA memory consumption. Models are trained with 8 NVIDIA GeForce RTX 2080 Ti GPUs. We save checkpoints every

[2] `https://pytorch.org/`

[3] `https://github.com/huggingface/pytorch-transformers`

Table 6.2: Main evaluation results. ‡ means statistically significant improvement over the strongest baseline with $p < 0.05$ tested by the Student's paired t-test. Methods in boldface are our models. "Q", "C", "S" denote previous **q**ueries, previous **c**licked and **s**kipped documents respectively.

Models	MRR	nDCG		
		@1	@3	@10
CARS[4] (Ahmad et al., 2019b)	0.4538	0.2940	0.4249	0.5109
BERT (Nogueira and Cho, 2019)	0.5198	0.3592	0.4984	0.5813
BERT-Concat-Q	0.5196	0.3596	0.4977	0.5806
BERT-Concat-QC	0.5340	0.3759	0.5149	0.5934
BERT-Concat-QCS	0.5366	0.3787	0.5174	0.5954
HBA-Transformers-QC	**0.5450**‡	**0.3866**‡	**0.5291**‡	**0.6021**‡
HBA-Transformers-QCS	0.5446‡	0.3850‡	0.5268‡	0.6012‡

5,000 steps and evaluate on 5,000 validation sessions (about 15% of the validation data) to select the best model for the test set. The training time for BERT-Concat and HBA-Transformers are 5.5 hours and 19.5 hours respectively.

6.4 Evaluation Results

In this section, we present our evaluation results, additional analyses on model ablation, and the conversation properties of queries.

6.4.1 Main Evaluation Results

We report the main evaluation results in Table 6.2. Note that the AOL data does not have ranking information and the skipped documents are synthetic as described in Section 6.3.1. We summarize our observations of the results as follows:

[4]We reproduce and re-evaluate the result of CARS with its open source code at `https://github.com/wasiahmad/context_attentive_ir`. The discrepancies in numbers come from different tie-breaking strategies in different evaluation methods. We use the official trec_eval while all models in the CARS paper are evaluated by an author-implemented function. With their evaluation, HBA-Transformers-QC gives the MRR of 0.6932 while CARS gives 0.542.

1. Even though CARS leverages the context information and is trained with multi-task learning, the BERT ranker *without* any context outperforms CARS. This improvement demonstrates the extraordinary capability of BERT in single-turn ranking.

2. BERT-Concat-QC(S) boosts the performance for BERT substantially. This suggests that history user behaviors can contribute to the ranking performance. In addition, it also shows that concatenating history turns in a BERT based model is effective despite its simplicity.

3. HBA-Transformers outperforms CARS by a large margin. Moreover, even though BERT-Concat is a very strong baseline, our method demonstrates statistically significant improvement over it with $p < 0.05$ tested by the Student's paired t-test. This shows the strength of our approach in modeling user behaviors. We present a detailed ablation analysis to investigate the sources of effectiveness in Section 6.4.2.

6.4.2 Ablation Analysis

Section 6.4.1 has shown the effectiveness of our model. This model performance is closely related to several design choices we made. In this section, we conduct an ablation analysis to investigate the contributions of each design choice, including the impact of the skipped documents, the history window size, the hierarchical behavior attention mechanism, and the behavior aware embeddings.

6.4.2.1 Impact of Skipped Documents and History Window Size

We investigate the impact of skipped documents and history window size in this section. The results are presented in Table 6.3.

Although we show that history user behaviors are informative in Section 6.4.1, we observe that providing different amount of history to the models does not show

Table 6.3: Impact of the skipped documents and history window size. We report MRR on the test set. Boldface denotes the best performance for each model.

Models	history=1	history=2	history=3
BERT-Concat-QC	**0.5376**	0.5345	0.5340
BERT-Concat-QCS	0.5343	0.5346	**0.5366**
HBA-Transformers-QC	**0.5593**	0.5429	0.5450
HBA-Transformers-QCS	0.5399	**0.5496**	0.5446

Table 6.4: Model ablation. ‡ and † means statistically significant performance *decrease* compared to the previous line with $p < 0.05$ and $p < 0.1$ tested by the Student's paired t-test.

Models	-QC	-QCS
Full Model	**0.5450**	**0.5446**
Without Behavior Aware Embeddings	0.5399‡	0.5432†
Without Hierarchical Behavior Attention	0.5340‡	0.5366‡

major differences. This could be explained by a property of the AOL data that many sessions are not made of a sequence of strictly evolving queries. More analysis on this property is presented in Section 6.4.3. In terms of the skipped documents, we see no consistent impacts. This could be due to the fact that the skipped document in the AOL data is synthetic instead of being recorded from real SERPs (Ahmad et al., 2019b). This is a limitation of our experiments and public available search logs.

6.4.2.2 Impact of Hierarchical Behavior Attention and Behavior Aware Embeddings

In this part, we further ablate our model to study the sources of effectiveness. Since the behavior aware embeddings is an integral part of the hierarchical behavior attention mechanism, our method without hierarchical behavior attention essentially is the same as the BERT-Concat model.

In Table 6.4, we observe that our models with behavior aware embeddings has better performance than those without, although the statistical significance is not

strong in the case of "-QCS". The hierarchical behavior attention, on the other hand, contribute statistically significant improvement to the model performance with and without the skipped documents. These results show that although behavior aware embeddings produce performance gains, the hierarchical architecture of the behavior attention mechanism is the primary source of effectiveness in our model.

6.4.3 Analysis of Conversation Properties

One of the motivations of our work is to lay the ground work for conversational search. Therefore, we analyze and characterize the conversational properties of queries to demonstrate the implications of our research in conversational search.

We use the MS GEN Encoder API[5] (Zhang et al., 2019) to encode the queries into hidden representations. Authors of GEN Encoder (Zhang et al., 2019) build a taxonomy for information-seeking behaviors, defined by the cosine similarity of the representations of query pairs as follows (Zhang et al., 2019):

- **`Topic Change`** (≤ 0.4): the two queries talk about different issues. E.g. "orlando theme parks" and "floral tattoo".
- **`Explore`** $(0, 4, 0.7]$: the second query explores around the intent of the first query. E.g. "river road camp" and "forest river rv".
- **`Specify`** $(0.7, 0.85]$: the second query drills down the intent of the first query. E.g. "basketball summer camps" and "rice university girls basketball summer camp".
- **`Paraphrase`** $(0.85, 1]$: the two queries share the exact same intent.

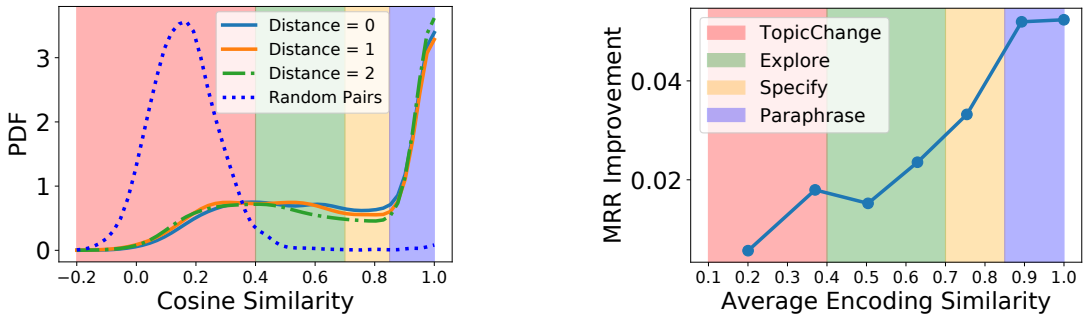
With these representations and the taxonomy, we conduct two parts of studies. In the first part, we follow the same setting as Zhang et al. (2019) to analyze the

[5]https://msturing.org/

distribution of cosine similarities between query pairs in search sessions. Queries within each pair can be adjacent, separated by one or two other queries in the session, or paired randomly. This result is presented in Figure 6.2a. We observe that random query pairs often talk about different topics as expected, while queries in the same session are much more related. Besides, for queries in the same session, many of them are similar to each other, despite having different distances. In comparison, Zhang et al. (2019) show that adjacent Bing queries follow a strong bimodal distribution, with less query pairs in extreme `Paraphrase`. Compared with Bing queries, AOL data has more paraphrasing queries and less queries that can form a long information seeking chain. This property of the data supports our finding in Section 6.4.2.1 that introducing different amount of session history has similar contribution to the performance. Our model performance on their Bing data could potentially benefit from a longer session history.

In the second part, we inspect the effect of our method on queries with different conversational properties. Given the current query, we compute the cosine similarity between the representations of this query and each previous query within the history window. These cosine similarity scores are averaged to provide a characterization of how *conversational* a query is. We adopt the same guideline to interpret the conversational property. From a session point of view, a session appears more *coherent* when it has more conversational queries that have similar intent. A session is not coherent if the intent of adjacent queries are constantly changing.

We analyze the correlation between the conversational property and the improvement of MRR. The MRR improvement is computed as the gain of our method in terms of the BERT baseline that is without session history. This improvement comes from the history user behaviors that are considered by our model (HBA-Transformers-QC in Table 6.2). The average encoding similarity scores are grouped by quantiles. We



(a) Distribution of CosSim

(b) MRR Improvement

Figure 6.2: Figure 6.2a shows the distributions of cosine similarities between queries in search sessions. The distance is the number of queries between the two queries. Query pairs were randomly sampled in the AOL data. Figure 6.2b shows the effect of our method on queries with different conversational properties. Taxonomy legends are shared.

merge the last four quantiles since their similarity scores are very similar. The results are shown in Figure 6.2b.

We observe that the MRR improvement gets progressively larger as the query become more conversational. For paraphrasing queries, the clicked/skipped documents could be different and thus can contribute to the performance. This result suggests that coherent sessions are more demanding in considering the history user behaviors in the session, and should be targeted by behavior aware models. Our HBA-Transformers can be potentially triggered at runtime upon identifying coherent sessions.

6.5 Summary

In this chapter, we introduce behavior awareness to a neural ranker for conversational document ranking. We propose a hierarchical behavior attention mechanism with behavior aware embeddings to let the system effectively distinguish and model different user behaviors. We show that a simple concatenation of history user behaviors is effective for a BERT based ranker. We further demonstrate that a behavior

aware hierarchical architecture is more powerful in this scenario than a simple concatenation. Moreover, we show that coherent sessions tends to be more conversational and thus are more demanding in considering history user behaviors.

CHAPTER 7

CLOSING REMARKS AND FUTURE WORK

7.1 Closing Remarks

Motivated by the iterative and interactive nature of conversational search, we study modeling conversation history in a conversational search process so that a conversational search system can better understand and fulfill the users' information needs. Many approaches we proposed take advantage of the recent advancement in pretrained language models. We start from history modeling for user intent prediction. We analyze the user intent patterns in information seeking conversations and investigate both feature-based methods and deep learning methods for user intent prediction. We then move to history modeling for conversational question answering, where we work from both history incorporation and history selection perspectives. After this, we continue this line of research in open-retrieval conversational question answering, where we emphasize the fundamental role of retrieval in conversational search. Finally, we study history modeling for conversational re-ranking and introduce behavior awareness to a neural ranker with a Hierarchical Behavior Aware Transformers model.

In Chapter 3, we analyze how people interact in information-seeking conversations as a crucial part in developing conversational search systems. We introduce the MSDialog dataset designed for this purpose and use it to analyze information-seeking conversations by user intent distribution, co-occurrence, and flow patterns. With MSDialog, we find some highly recurring patterns in user intent during an information-seeking process. They could be useful for designing conversational search systems.

Furthermore, we study how to predict user intent in information-seeking conversations from two aspects. First, we extract features based on the content, structural, and sentiment characteristics of a given utterance, and use classic machine learning methods to perform user intent prediction. We then conduct an in-depth feature importance analysis to identify key features in this prediction task. We find that structural features contribute most to the prediction performance. Given this finding, we construct neural classifiers to incorporate context information and achieve better performance without feature engineering. Our findings can provide insights into the important factors and effective methods of user intent prediction in information-seeking conversations.

In Chapter 4, we study the modeling of conversation history for conversational question answering. We propose a novel solution for ConvQA that involves three aspects. First, we propose a history answer embedding method to encode the conversation history using BERT (Devlin et al., 2019) in a natural way. We further introduce the position information of a question into history answer embedding. Second, we design a history attention mechanism to conduct a "soft selection" for conversation history turns. This method attends to history turns with different weights based on how helpful they are on answering the current question. Third, in addition to handling conversation history, we take advantage of multi-task learning to do answer prediction along with dialog act prediction using a uniform model architecture. We demonstrate the effectiveness of our model with extensive experimental evaluations on QuAC, a large-scale ConvQA dataset. We show that our positional history answer embedding approach is highly effective, especially for BERT-like models whose maximum input sequence length is limited. We further discover that position information plays an important role in conversation history modeling. We also visualize the history attention and provide new insights into conversation history understanding.

In Chapter 5, we address the limitations of Chapter 4 by introducing an open-retrieval conversational question answering 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 build an end-to-end system for ORConvQA, featuring a retriever, a reranker, and a reader that are all based on Transformers. We first conduct experiments in a fully-supervised setting. Our extensive experiments 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. We then dig into a weakly-supervised setting to tackle the freeform answers in information-seeking conversations. Different from span answers, freeform answers are not necessarily strict spans of any passage. We introduce a learned weak supervision approach that can identify a paraphrased span of the known answer in a passage. Our experiments show that a span-match weak supervisor can only handle conversations with span answers, and has less satisfactory results for freeform answers generated by people. Our method is more flexible as it can handle both span answers and freeform answers. Moreover, our method can be more powerful when combined with the span-match method, which shows it is complementary to the span-match method.

Finally, in Chapter 6, we focus on the contextual document ranking task, which deals with the challenge of user interaction history modeling for conversational search. Given a history of user feedback behaviors, such as issuing a query, clicking a document, and skipping a document, we propose to introduce behavior awareness to a neural ranker, resulting in a Hierarchical Behavior Aware Transformers model. The hierarchy is composed of an intra-behavior attention layer and an inter-behavior attention layer to let the system effectively distinguish and model different user behaviors. Our experiments demonstrate that the hierarchical behavior aware architecture is more powerful than a simple combination of history behaviors. In addition, we an-

alyze the conversational property of queries. We show that coherent sessions tend to be more conversational and thus are more demanding in terms of considering history user behaviors.

7.2 Future Work

This dissertation presents our initial effort on modeling conversation history for conversational IR. There are still many challenges in this area that we would like to study in the future.

- **User Intent Prediction**. Instead of predicting user intent on an utterance level, a more fine-grained intent detection approach might be desired to reveal users' information needs more accurately. Furthermore, we would like to study how to leverage the predicted user intent to retrieve, rank, or generate conversational responses, including both answers and clarifying questions, in an information-seeking setting.

- **Conversational Question Answering**. Related to the previous direction, we would like to further analyze the relationship between history attention patterns and different user intents or dialog acts. This could lead to a history selection or weighting scheme guided by user intent in a more explicit manner. Another direction we would like to pursue is generating natural answers based on the predicted answer span.

- **Open-Retrieval Conversational Question Answering**. Following our other efforts described in Section 5.2.5, we would like to continue working on a retriever that is not only learnable but also tunable by the downstream ConvQA task. In addition, we will also investigate other effective approaches to conduct post-domain pretraining for the language models used in conversational search systems. Finally, combining the power of sparse retrieval and dense retrieval

for a hybrid approach should also be a promising direction in open-retrieval ConvQA.

- **Conversational Re-Ranking**. A diverse set of user behaviors could better inform the history modeling process. Therefore, in future work, we will consider using more diverse user behaviors that are beyond queries, clicks, and skips.

BIBLIOGRAPHY

Eugene Agichtein, Eric Brill, Susan Dumais, and Robert Ragno. Learning user interaction models for predicting web search result preferences. In *SIGIR*, 2006.

Amin Ahmad, Noah Constant, Yinfei Yang, and Daniel Cer. ReQA: An evaluation for end-to-end answer retrieval models. *ArXiv*, 2019a.

Wasi Uddin Ahmad, Kai Wei Chang, and Hongning Wang. Multi-task learning for document ranking and query suggestion. In *ICLR*, 2018.

Wasi Uddin Ahmad, Kai Wei Chang, and Hongning Wang. Context attentive document ranking and query suggestion. In *SIGIR*, 2019b.

Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. Asking Clarifying Questions in Open-Domain Information-Seeking Conversations. In *SIGIR*, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization. *ArXiv*, 2016.

Nicholas J. Belkin, Colleen Cool, Adelheit Stein, and Ulrich Thiel. Cases, scripts, and information-seeking strategies: On the design of interactive information retrieval systems. *Expert Systems With Applications*, 1995.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The Long-Document Transformer. *ArXiv*, 2020.

Paul N. Bennett, Ryen W. White, Wei Chu, Susan T. Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. Modeling the impact of short- and long-term behavior on search personalization. In *SIGIR*, 2012.

Sumit Bhatia, Prakhar Biyani, and Prasenjit Mitra. Classifying User Messages For Managing Web Forum Data. In *WebDB*, 2012.

Sumit Bhatia, Prakhar Biyani, and Prasenjit Mitra. Summarizing online forum discussions - Can dialog acts of individual messages help? In *EMNLP*, 2014.

Keping Bi, Qingyao Ai, Yongfeng Zhang, and W. Bruce Croft. Conversational product search based on negative feedback. In *CIKM*, 2019.

Jean Carletta, Stephen Isard, Gwyneth Doherty-Sneddon, Amy Isard, Jacqueline C. Kowtko, and Anne H. Anderson. The reliability of a dialogue structure coding scheme. *Comput. Linguist.*, 1997.

Ben Carterette, Paul Clough, Mark Hall, Evangelos Kanoulas, and Mark Sanderson. Evaluating retrieval over sessions: The TREC session track 2011-2014. In *SIGIR*, 2016.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to answer open-domain questions. In *ACL*, 2017.

Jiecao Chen, Liu Yang, Karthik Raman, Michael Bendersky, Jung-Jung Yeh, Yun Zhou, Marc Najork, Danyang Cai, and Ehsan Emadzadeh. DiPair: Fast and Accurate Distillation for Trillion-Scale Text Matching and Pair Modeling. In *EMNLP*, 2020.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. GraphFlow: Exploiting Conversation Flow with Graph Neural Networks for Conversational Machine Comprehension. *ArXiv*, 2019a.

Zhipeng Chen, Yiming Cui, Wentao Ma, Shijin Wang, and Guoping Hu. Convolutional Spatial Attention Model for Reading Comprehension with Multiple-Choice Questions. In *AAAI*, 2019b.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. QuAC: Question Answering in Context. In *EMNLP*, 2019.

Aleksandr Chuklin, Aliaksei Severyn, Johanne R. Trippas, Enrique Alfonseca, Hanna Silen, and Damiano Spina. Using audio transformations to improve comprehension in voice question answering. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *ICLR*, 2020.

Daniel Cohen and W. Bruce Croft. End to End Long Short Term Memory Networks for Non-Factoid Question Answering. In *ICTIR*, 2016.

Daniel Cohen, Liu Yang, and W. Bruce Croft. WikiPassageQA: A Benchmark Collection for Research on Non-factoid Answer Passage Retrieval. In *SIGIR*, 2018.

W. Bruce Croft and R. H. Thompson. I3R: A New Approach to the Design of Document Retrieval Systems. *Journal of the American Society for Information Science*, 1987.

Guoxin Cui and Xueqi Cheng. ICTNET at Session Track TREC2014. In *TREC*, 2014.

Zhuyun Dai and Jamie Callan. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *SIGIR*, 2019.

Zhuyun Dai, Cenyan Xiong, Jamie P. Callan, and Zhiyuan Liu. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search. In *WSDM*, 2018.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. Multi-step Retriever-reader Interaction for Scalable Open-domain Question Answering. In *ICLR*, 2019.

Debajyoti Datta, Valentina Brashers, John Owen, Casey White, and Laura E. Barnes. A Deep Learning Methodology for Semantic Utterance Classification in Virtual Human Dialogue Systems. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, 2019.

Bhuwan Dhingra, Kathryn Mazaitis, and William W. Cohen. Quasar: Datasets for Question Answering by Search and Reading. *ArXiv*, 2017. URL `http://arxiv.org/abs/1707.03904`.

Shilin Ding, Gao Cong, Chin Yew Lin, and Xiaoyan Zhu. Using Conditional Random Fields to Extract Contexts and Answers of Questions from Online Forums. In *ACL*, 2008.

Susan Dumais, Edward Cutrell, Jonathan J. Cadiz, Gavin Jancke, Raman Sarin, and Daniel C. Robbins. Stuff I've Seen: A System for Personal Information Retrieval and Re-use. In *SIGIR*, 2003.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Guney, Volkan Cirik, and Kyunghyun Cho. SearchQA: A New Q&A Dataset Augmented with Context from a Search Engine. *ArXiv*, 2017.

Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. Can You Unpack That? Learning to Rewrite Questions-in-Context. In *EMNLP/IJCNLP*, pages 5917–5923, 2019.

Jianfeng Gao, Xiaodong He, and Jian Yun Nie. Clickthrough-based Translation Models for Web Search: From Word Models to Phrase Models. In *International Conference on Information and Knowledge Management, Proceedings*, 2010.

Matt Gardner and Christopher Clark. Simple and Effective Multi-paragraph Reading Comprehension. In *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 2018.

John J. Godfrey and Edward Holliman. Switchboard-1 Release 2. *Linguistic Data Consortium, Philadelphia*, 1997.

Alex Graves and Jürgen Schmidhuber. Framewise Phoneme Classification with Bidirectional LSTM Networks. *Proceedings of the International Joint Conference on Neural Networks*, 2005.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *CIKM*, 2016.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In *ACL*, 2020.

Kelvin Guu, Kenton Lee, Z. Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-Augmented Language Model Pre-Training. *ArXiv*, 2020.

Matthias Hagen, Michael Völske, Jakob Gomoll, Marie Bornemann, Lene Ganschow, Florian Kneist, Abdul Hamid Sabri, and Benno Stein. Webis at TREC 2013 Sessions and Web Track. In *TREC*, 2013.

Helia Hashemi, Hamed Zamani, and W. Croft. Guided Transformer: Leveraging Multiple External Sources for Representation Learning in Conversational Search. In *SIGIR*, 2020.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. The Third Dialog State Tracking Challenge. In *SLT*, 2014.

Dan Hendrycks and Kevin Gimpel. Gaussian Error Linear Units (GELUs). *ArXiv*, 2016.

Chiori Hori and Takaaki Hori. End-to-end Conversation Modeling Track in DSTC6. *ArXiv*, 2017.

Phu Mon Htut, Samuel Bowman, and Kyunghyun Cho. Training a Ranking Function for Open-Domain Question Answering. In *NAACL-HLT*, 2018.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *NIPS*, 2014.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. Reinforced Mnemonic Reader for Machine Reading Comprehension. In *IJCAI*, 2018.

Hsin Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. FusionNet: Fusing via Fully-aware Attention with Application to Machine Comprehension. In *ICLR*, 2018a.

Hsin Yuan Huang, Wen Tau Yih, and Eunsol Choi. FlowQA: Grasping Flow in History for Conversational Machine Comprehension. *ICLR*, 2019.

Jizhou Huang, Wei Zhang, Yaming Sun, Haifeng Wang, and Ting Liu. Improving Entity Recommendation with Search Log and Multi-task Learning. In *IJCAI*, 2018b.

Po Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *CIKM*, 2013.

C. J. Hutto and Eric Gilbert. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. In *ICWSM*, 2014.

Mohit Iyyer, Jordan L. Boyd-Graber, Leonardo Claudino, R. Socher, and Hal Daumé. A Neural Network for Factoid Question Answering over Paragraphs. In *EMNLP*, 2014.

Kalervo Järvelin, Susan L. Price, Lois M.L. Delcambre, and Marianne Lykke Nielsen. Discounted Cumulated Gain Based Evaluation of Multiple-Query IR Sessions. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008.

Zhuolin Jiang, A. El-Jaroudi, W. Hartmann, D. Karakos, and L. Zhao. Cross-lingual Information Retrieval with BERT. In *CLSSTS@LREC*, 2020.

Jeff Johnson, Matthijs Douze, and Herve Jegou. Billion-scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, pages 1–1, 2019. ISSN 2332-7790. doi: 10.1109/tbdata.2019.2921572.

Rosie Jones and Kristina Lisa Klinkner. Beyond the Session Timeout: Automatic Hierarchical Segmentation of Search Topics in Query Logs. In *CIKM*, 2008.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *ACL*, 2017.

Evangelos Kanoulas, Ben Carterette, Paul D. Clough, and Mark Sanderson. Evaluating multi-query sessions. In *SIGIR*, 2011.

Vladimir Karpukhin, Barlas Ouguz, Sewon Min, Ledell Yu Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense Passage Retrieval for Open-Domain Question Answering. *ArXiv*, abs/2004.04906, 2020.

Hamed Khanpour, Nishitha Guntakandla, and Rodney Nielsen. Dialogue Act Classification in Domain-independent Conversations Using a Deep Recurrent Neural Network. In *COLING*, 2016.

Simran Khanuja, S. Dandapat, A. Srinivasan, Sunayana Sitaram, and M. Choudhury. GLUECoS : An Evaluation Benchmark for Code-Switched NLP. In *ACL*, 2020.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *SIGIR*, 2020.

Seokhwan Kim, Luis Fernando D'Haro, Rafael E. Banchs, Jason D. Williams, Matthew Henderson, and Koichiro Yoshino. The Fifth Dialog State Tracking Challenge. In *SLT*, 2017.

Yoon Kim. Convolutional Neural Networks for Sentence Classification. In *EMNLP*, 2014.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.

Andrei P. Kirilenko and Svetlana Stepchenkova. Inter-coder Agreement in One-to-many Classification: Fuzzy Kappa. *PLoS ONE*, 11(3), 2016. ISSN 19326203. doi: 10.1371/journal.pone.0149787.

Nikita Kitaev, L. Kaiser, and Anselm Levskaya. Reformer: The Efficient Transformer. In *ICLR*, 2020.

Bernhard Kratzwald and Stefan Feuerriegel. Adaptive Document Retrieval for Deep Question Answering. In *EMNLP*, 2019.

Kalpesh Krishna, John Wieting, and Mohit Iyyer. Reformulating Unsupervised Style Transfer as Paraphrase Generation. In *EMNLP*, 2020.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural Questions: A Benchmark for Question Answering Research. *TACL*, 2019.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *ArXiv*, 2019.

Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. Ranking Paragraphs for Improving Answer Recall in Open-Domain Question Answering. In *EMNLP*, 2019a.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent Retrieval for Weakly Supervised Open Domain Question Answering. In *ACL*, 2019b.

Yen-Chieh Lien, Hamed Zamani, and W. Croft. Recipe Retrieval with Visual Query of Ingredients. In *SIGIR*, 2020.

Constantine Lignos, Daniel Cohen, Yen-Chieh Lien, Pratik Mehta, W. Bruce Croft, and Scott Miller. The Challenges of Optimizing Machine Translation for Low Resource Cross-Language Information Retrieval. In *EMNLP/IJCNLP*, 2019.

Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion Observer. In *WWW*, 2005.

Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye Yi Wang. Representation Learning Using Multi-task Deep Neural Networks for Semantic Classification and Information Retrieval. In *NAACL-HLT*, 2015.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-Task Deep Neural Networks for Natural Language Understanding. *ArXiv*, 2019a.

Yang Liu and Mirella Lapata. Hierarchical Transformers for Multi-Document Summarization. In *ACL*, 2019.

Yang Liu, Kun Han, Zhao Tan, and Yun Lei. Using Context Information for Dialog Act Classification in DNN Framework. In *EMNLP*, 2018.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *ArXiv*, 2019b.

Zhuang Liu, Kaiyu Huang, Degen Huang, and Jun Zhao. Dual Head-wise Coattention Network for Machine Comprehension with Multiple-Choice Questions. In *CIKM*, 2020.

Ryan Lowe, Nissan Pow, Iulian V. Serban, and Joelle Pineau. The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-turn Dialogue Systems. In *SIGDIAL*, 2015.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. Sparse, Dense, and Attentional Representations for Text Retrieval. *ArXiv*, 2020.

Sean MacAvaney, Arman Cohan, Andrew Yates, and Nazli Goharian. CEDR: Contextualized Embeddings for Document Ranking. In *SIGIR*, 2019.

Gary Marchionini. Exploratory Search: From Finding to Understanding. *Communications of the ACM*, 2006.

Kenneth Marino, M. Rastegari, Ali Farhadi, and R. Mottaghi. OK-VQA: A Visual Question Answering Benchmark Requiring External Knowledge. In *CVPR*, 2019.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to Match Using Local and Distributed Representations of Text for Web Search. In *26th International World Wide Web Conference, WWW 2017*, 2017.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A Human Generated Machine Reading Comprehension Dataset. *CEUR Workshop Proceedings*, 2016.

Rodrigo Nogueira and Kyunghyun Cho. Passage Re-ranking with BERT. *ArXiv*, 2019. URL `http://arxiv.org/abs/1901.04085`.

R. N. Oddy. *Information Retrieval Through Man-machine Dialogue*. Emerald Group Publishing Limited, 1977.

Andrew Olney, Max Louwerse, Eric Matthews, Johanna Marineau, Heather Hite-Mitchell, and Arthur Graesser. Utterance Classification in AutoTutor. In *HLT-NAACL-EDUC*, 2003.

Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A picture of search. In *ACM International Conference Proceeding Series*, 2006.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. In *EMNLP*, 2014.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations. In *NAACL-HLT*, 2018.

Chen Qu, Feng Ji, Minghui Qiu, Liu Yang, Zhiyu Min, Haiqing Chen, Jun Huang, and W. Bruce Croft. Learning to Selectively Transfer: Reinforced Transfer Learning for Deep Text Matching. In *WSDM*, 2019a.

Chen Qu, Liu Yang, W. Bruce Croft, Falk Scholer, and Yongfeng Zhang. Answer Interaction in Non-factoid Question Answering Systems. In *CHIIR*, 2019b.

Chen Qu, Liu Yang, Minghui Qiu, W. Bruce Croft, Yongfeng Zhang, and Mohit Iyyer. BERT with History Answer Embedding for Conversational Question Answering. In *SIGIR*, 2019c.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, 2019.

Filip Radlinski and Nick Craswell. A Theoretical Framework for Conversational Search. In *CHIIR*, 2017.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*, 2016.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know What You Don't Know: Unanswerable Questions for SQuAD. In *ACL*, 2018.

Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A Conversational Question Answering Challenge. *TACL*, 2019.

Alan Ritter, Colin Cherry, and Bill Dolan. Unsupervised Modeling of Twitter Conversations. In *NAACL-HLT*, 2010.

Alan Ritter, Colin Cherry, and William B. Dolan. Data-driven Response Generation in Social Media. In *EMNLP*, 2011.

Stephen Robertson and Hugo Zaragoza. The Probabilistic Relevance Framework: BM25 and Beyond. In *Foundations and Trends in Information Retrieval*, 2009.

Abigail See, Peter J. Liu, and Christopher D. Manning. Get To The Point: Summarization with Pointer-Generator Networks. In *ACL*, 2017.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional Attention Flow for Machine Comprehension. In *ICLR*, 2016.

Chirag Shah and Jefferey Pomerantz. Evaluating and predicting answer quality in community QA. In *SIGIR*, 2010.

Xuehua Shen, Bin Tan, and Chengxiang Zhai. Context-Sensitive Information Retrieval Using Implicit Feedback. In *SIGIR*, 2005.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning Semantic Representations Using Convolutional Neural Networks for Web Search. In *WWW*, 2014.

Sosuke Shiga, Hideo Joho, Roi Blanco, Johanne R. Trippas, and Mark Sanderson. Modelling Information Needs in Collaborative Search Conversations. In *SIGIR*, 2017.

Anshumali Shrivastava and Ping Li. Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS). In *NIPS*, 2014.

Philipp Sorg and P. Cimiano. Exploiting Wikipedia for Cross-lingual and Multilingual Information Retrieval. *Data Knowl. Eng.*, pages 26–45, 2012.

Andreas Stolcke, Noah Coccaro, Rebecca Bates, Paul Taylor, Carol Van Ess-Dykema, Klaus Ries, Elizabeth Shriberg, Daniel Jurafsky, Rachel Martin, and Marie Meteer. Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. *Computational Linguistics*, 2000.

Dinoj Surendran and Gina Anne Levow. Dialog Act Tagging with Support Vector Machines and Hidden Markov Models. In *INTERSPEECH/ICSLP*, 2006.

Hao Tan and M. Bansal. LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In *EMNLP/IJCNLP*, 2019.

Computing Technology, Chinese Academy, and Session Track. ICTNET at Session Track TREC 2013. In *TREC*, 2013.

Paul Thomas, Daniel McDuff, Mary Czerwinski, and Nick Craswell. MISC: A Data Set of Information-seeking Conversations. In *ProQuest Dissertations and Theses*, 2017.

Johanne R. Trippas, Damiano Spina, Lawrence Cavedon, and Mark Sanderson. How Do People Interact in Conversational Speech-only Search Tasks: A Preliminary Analysis. In *CHIIR*, 2017.

Johanne R. Trippas, Damiano Spina, Lawrence Cavedon, Hideo Joho, and Mark Sanderson. Informing the Design of Spoken Conversational Search. In *CHIIR*, 2018.

Johanne R. Trippas, Damiano Spina, Paul Thomas, Mark Sanderson, Hideo Joho, and Lawrence Cavedon. Towards a Model for Spoken Conversational Search. *Information Processing and Management*, 2020.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models. *ArXiv*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *NIPS*, 2017.

Ellen M Voorhees and Dawn M Tice. The TREC-8 Question Answering Track Evaluation. In *TREC*, 1999.

Hongning Wang, Yang Song, Ming Wei Chang, Xiaodong He, Ryen W. White, and Wei Chu. Learning to extract cross-session search tasks. In *WWW*, 2013.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA. In *EMNLP-CoNLL*, 2007.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. R3: Reinforced Ranker-Reader for Open-Domain Question Answering. In *AAAI*, 2018.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-Attention with Linear Complexity. *ArXiv*, 2020.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated Self-matching Networks for Reading Comprehension and Question Answering. In *ACL*, 2017.

Ryen W. White and Resa A. Roth. Exploratory Search: Beyond the Query-Response Paradigm. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 2009.

Ryen W. White, Paul N. Bennett, and Susan T. Dumais. Predicting Short-term Interests Using Activity-based Search Context. In *CIKM*, 2010.

Ryen W. White, Wei Chu, Ahmed Hassan, Xiaodong He, Yang Song, and Hongning Wang. Enhancing personalized search by mining and modeling task behavior. In *WWW*, 2013.

John Wieting and Kevin Gimpel. ParaNMT-50M: Pushing the Limits of Paraphrastic Sentence Embeddings with Millions of Machine Translations. In *ACL*, 2018.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. TransferTransfo: A Transfer Learning Approach for Neural Network Based Conversational Agents. In *NeurIPS CAI Workshop*, 2018.

Yu Wu, Wei Yu Wu, Ming Zhou, and Zhoujun Li. Sequential Match Network: A New Architecture for Multi-turn Response Selection in Retrieval-based Chatbots. In *ACL*, 2016.

Biao Xiang, Daxin Jiang, Jian Pei, Xiaohui Sun, Enhong Chen, and Hang Li. Context-aware Ranking in Web Search. In *SIGIR*, 2010.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end Neural Ad-hoc Ranking with Kernel Pooling. In *SIGIR*, 2017.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, J. Liu, P. Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. *ArXiv*, 2020.

Yichong Xu, Xiaodong Liu, Yelong Shen, Jingjing Liu, and Jianfeng Gao. Multi-Task Learning for Machine Reading Comprehension. *ArXiv*, 2018.

Zhao Yan, Nan Duan, Peng Chen, Ming Zhou, Jianshe Zhou, and Zhoujun Li. Building Task-oriented Dialogue Systems for Online Shopping. In *AAAI*, 2017.

Liu Yang, Qingyao Ai, Jiafeng Guo, and W. Bruce Croft. aNMM: Ranking Short Answer Texts with Attention-based Neural Matching Model. In *CIKM*, 2016.

Liu Yang, Hamed Zamani, Yongfeng Zhang, Jiafeng Guo, and W. Bruce Croft. Neural Matching Models for Question Retrieval and Next Question Prediction in Conversation. *ArXiv*, 2017.

Liu Yang, Minghui Qiu, Chen Qu, Jiafeng Guo, Yongfeng Zhang, W. Bruce Croft, Jun Huang, and Haiqing Chen. Response Ranking with Deep Matching Networks and External Knowledge in Information-seeking Conversation Systems. In *SIGIR*, 2018.

Liu Yang, Junjie Hu, Minghui Qiu, Chen Qu, Jianfeng Gao, W Bruce Croft, Xiaodong Liu, Yelong Shen, and Jingjing Liu. A Hybrid Retrieval-Generation Neural Conversation Model. In *CIKM*, 2019a.

Liu Yang, Minghui Qiu, Chen Qu, Cen Chen, Jiafeng Guo, Yongfeng Zhang, W. Bruce Croft, and Haiqing Chen. IART: Intent-aware Response Ranking with Transformers in Information-seeking Conversation Systems. In *WWW*, 2020a.

Liu Yang, Mingyang Zhang, Cheng Li, Mike Bendersky, and Marc Najork. Beyond 512 Tokens: Siamese Multi-depth Transformer-based Hierarchical Encoder for Long-Form Document Matching. In *CIKM*, 2020b.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-End Open-Domain Question Answering with BERTserini. In *NAACL-HLT*, 2019b.

Yi Yang, Wen Tau Yih, and Christopher Meek. WikiQA: A Challenge Dataset for Open-domain Question Answering. In *EMNLP*, 2015.

Z. Yang, Zihang Dai, Yiming Yang, J. Carbonell, R. Salakhutdinov, and Quoc V. Le. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NeurIPS*, 2019c.

Mark Yatskar. A Qualitative Comparison of CoQA, SQuAD 2.0 and QuAC. In *NAACL-HLT*, 2018.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In *ICLR*, 2018.

Yue Yu, Siyao Peng, and Grace Hui Yang. Modeling Long-Range Context for Concurrent Dialogue Acts Recognition. In *CIKM*, 2019.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, C. Alberti, S. Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for Longer Sequences. In *NeurIPS*, 2020.

Hamed Zamani, Bhaskar Mitra, Xia Song, Nick Craswell, and Saurabh Tiwary. Neural Ranking Models with Multiple Document Fields. In *WSDM*, 2017.

Hamed Zamani, Susan T. Dumais, Nick Craswell, Paul Bennett, and Gord Lueck. Generating Clarifying Questions for Information Retrieval. In *WWW*, 2020.

Hongfei Zhang, Xia Song, Chenyan Xiong, Corby Rosset, Paul N. Bennett, Nick Craswell, and Saurabh Tiwary. Generic Intent Representation in Web Search. In *SIGIR*, 2019.

Shuo Zhang, Zhuyun Dai, Krisztian Balog, and Jamie Callan. Summarizing and Exploring Tabular Data in Conversational Search. In *SIGIR*, 2020.

Xiang Zhang, Junbo Zhao, and Yann Lecun. Character-level Convolutional Networks for Text Classification. In *NIPS*, 2015.

Xiaodong Zhang and Houfeng Wang. A Joint Model of Intent Determination and Slot Filling for Spoken Language Understanding. In *IJCAI*, 2016.

Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W. Bruce Croft. Towards Conversational Search and Recommendation: System Ask, User Respond. In *CIKM*, 2018.

Yukun Zheng, Zhen Fan, Yiqun Liu, Cheng Luo, Min Zhang, and Shaoping Ma. Sogou-QCL: A New Dataset with Click Relevance Label. In *SIGIR*, 2018.

Chenguang Zhu, Michael Zeng, and Xuedong Huang. SDNet: Contextualized Attention-based Deep Network for Conversational Question Answering. *ArXiv*, 2018.