Teaching Machines to Ask and Answer Questions: Knowledge Assessment and Information Acquisition in Reading Comprehension

GAO, Yifan

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
September 2021

Thesis Assessment Committee

Abstract of thesis entitled:

Teaching Machines to Ask and Answer Questions:

Knowledge Assessment and Information Acquisition in Reading Comprehension

Submitted by GAO, Yifan

for the degree of Doctor of Philosophy

at The Chinese University of Hong Kong in September 2021

The capacity to use complex language to communicate and maintain our social world has been a trademark of humanity. Among different goals of language usage, questions and answers play significant roles in our day-to-day communications. We ask questions to explore unknown information from our side, and we answer questions to fill the information gap of others. As core skills to pass the Turing test, teaching machines to ask and answer questions are fundamental while challenging natural language processing tasks and have long been associated with artificial intelligence. In this thesis, we teach machines to ask and answer reading comprehension questions towards a passage of text. We focus on two crucial goals in question asking and answering: knowledge assessment and information acquisition. On the one hand, asking questions in reading comprehension passages is the key step to generate knowledge assessment exercises for educational purposes. On the other hand, many high-level questions are inherently underspecified in our day-to-day conversations. The machine needs to keep asking clarification questions until it gathers enough information to answer those high-level questions. The contributions of this thesis are grouped into five parts. As described in the following, the former three focus on question generation for knowledge assessment while the latter two investigate asking and answering questions for information acquisition.

First, we investigate the difficulty levels of questions in reading comprehension and propose a new question generation setting, named Difficulty-controllable Question Generation (DQG). Taking as input a sentence in the reading comprehension paragraph and some of its text fragments (i.e., answers) that we want to ask questions about, a DQG method needs to generate questions under the control of specified difficulty labels. We propose an end-to-end framework with difficulty-aware proximity hints and a difficulty-controllable question decoder. On our prepared first dataset of reading comprehension questions with difficulty labels, the results show that the question generated by our framework not only have better quality under the metrics like BLEU but also comply with the specified difficulty labels.

Second, we investigate the task of distractor generation for multiple-choice reading comprehension questions from real examinations. Taking a reading comprehension article, a pair of question and its correct option as input, our goal is to generate several distractors which are somehow related to the answer, consistent with the semantic context of the question and have some trace in the article. We propose a hierarchical encoder-decoder framework with static and dynamic attention mechanisms to tackle this task. The proposed framework outperforms several strong baselines on the first prepared distractor generation dataset of real reading comprehension questions.

Third, we study the problem of generating interconnected questions in question-answering style conversations. In a coherent conversation, questions are highly conversational and have smooth transitions between turns. We propose an end-to-end neural model with coreference alignment and conversation flow modeling. Automatic and human evaluations show that our system outperforms several baselines and can generate highly conversational questions.

Fourth, we focus on conversational machine reading. Machines can take the initiative to ask users questions that help to solve their problems instead of jumping to a conclusion hurriedly. We present a new framework of conversational machine reading that comprises a novel Explicit Memory Tracker (EMT) to track whether conditions listed in the rule text have already been satisfied to make a decision. Moreover, our framework generates clarification questions by adopting a coarse-to-fine reasoning strategy. EMT outperforms existing methods as well as gains interpretability by visualizing the entailment-oriented reasoning process as the conversation flows.

Finally, we propose Discern, a discourse-aware entailment reasoning network to strengthen the connection and enhance the understanding of both document and dialog in conversational machine reading. Specifically, we split the document into clause-like elementary discourse units (EDU) using a pre-trained discourse segmentation model. Then we train our model in a weakly supervised manner to predict whether the user feedback in a conversation entails each EDU. Based on the learned EDU and entailment representations, we either reply to the user our final decision "yes/no/irrelevant" of the initial question or generate a follow-up question to inquiry more information.

Discern achieves state-of-the-art performance in conversational machine reading.

論文題目：教機器提問和回答問題：閱讀理解中的知識評估與信息獲取

作者：高一帆

學校：香港中文大學

學系：計算機科學與工程學系

修讀學位：哲學博士

摘要：

使用複雜語言來交流和維護我們的社交世界的能力一直是人類的標誌。在語言使用的不同目標中，問答在我們的日常交流中扮演著重要的角色。我們提問是為了探索我們身邊的未知信息，我們回答問題是為了填補他人的信息空白。作為通過圖靈測試的核心技能，教機器提問和回答問題是基礎但充滿挑戰的自然語言處理任務，並且長期以來一直與人工智能相關聯。在本論文中，我們教機器針對一段文本提出和回答閱讀理解問題。我們專注於提問和回答的兩個關鍵目標：知識評估和信息獲取。一方面，在閱讀理解文章中提出問題是為教育目的生成知識評估練習的關鍵步驟。另一方面，在我們的日常對話中，許多問

題具有模糊性。機器需要不斷提出澄清問題，直到它收集到足夠的信息來回答這些問題。本論文的貢獻分為五個部分。如下所述，前三者側重於知識評估中的問題生成，而後兩者則研究信息獲取中的提問和回答問題。

首先，我們調查了閱讀理解問題的難度級別，並提出了一種新的問題生成設定，稱為難度可控問題生成（DQG）。將閱讀理解段落中的一個句子及其一些我們想要提問的文本片段（即答案）作為輸入，DQG 方法需要在指定難度標籤的控制下生成問題。我們提出了一個端到端框架，該框架具有難度感知鄰近提示和難度可控問題解碼器。在我們準備的帶有難度標籤的閱讀理解問題的第一個數據集上，結果表明，我們的框架生成的問題不僅在 BLEU 等指標下具有更好的質量，而且符合指定的難度標籤。

其次，我們研究了真實考試中多項選擇閱讀理解問題的干擾項生成任務。以一篇閱讀理解文章、一對問題及其正確選項作為輸入，我們的目標是生成幾個乾擾項，這些干擾項與答案有某種關係，與問題的語義上下文一致，並在文章中有一些痕跡。我們提出了一個具有靜態和動態注意機制的分層編碼器-解碼器框架來解決這個任務。

所提出的框架在第一個準備好的真實閱讀理解問題的干擾生成數據集上優於幾個強大的基線。

第三，我們研究了在問答式對話中產生相互關聯的問題的問題。在連貫的對話中，問題是高度對話的，並且在輪次之間有平滑的過渡。我們提出了一種具有共指對齊和對話流建模的端到端神經模型。自動和人工評估表明，我們的系統優於多個基線，並且可以生成高度對話的問題。

第四，我們專注於會話式機器閱讀。機器可以主動向用戶提出有助於解決用戶問題的澄清問題，而不是倉促下結論。我們提出了一種新的對話機器閱讀框架，該框架包括一個新穎的顯式內存跟踪器（EMT），用於跟踪是否已經滿足規則文本中列出的條件以做出決定。此外，我們的框架通過採用從粗到精的推理策略來生成澄清問題。EMT 優於現有方法，並通過在對話流中可視化面向蘊涵的推理過程來獲得可解釋性。

最後，我們提出了 Discern，這是一個話語感知的蘊涵推理網絡，以加強連接並增強對會話機器閱讀中文檔和對話的理解。具體來說，我們使用預訓練的話語分割模型將文檔拆分為類似子句的基本話語單元（EDU）。然後我們以弱監督的方

式訓練我們的模型，以預測對話中的用戶反饋是否需要每個 EDU。基於學習到的 EDU 和蘊含表示，我們要么回復用戶我們對初始問題的最終決定“是/否/不相關”，要么生成後續問題以查詢更多信息。 Discern 在對話式機器閱讀方面達到了最先進的性能。

Contents

Abstract **i**

1 Introduction **1**

1.1 Motivation 1

1.2 Contributions 6

1.3 Thesis Outline 10

2 Background Review **12**

2.1 Deep Learning Basics for NLP 13

2.1.1 RNN and Language Models 14

2.1.2 Sequence-to-Sequence Models 15

2.1.3 Transformer and Pretraining 17

2.1.4 Evaluation Metrics 19

2.2 Question Generation 21

2.2.1 Textual QG 21

2.2.2 Dialogue QG 23

2.3 Machine Reading Comprehension 24

2.3.1 Textual MRC 25

2.3.2 Dialogue MRC 26

2.4 Other QA Tasks 29
2.4.1 Knowledge-Based Question Answering . . 29
2.4.2 Community Question Answering 30

3 Difficulty Controllable Question Generation 33
3.1 Introduction 33
3.2 Task Definition 37
3.3 The Protocol of Difficulty Labeling 38
3.4 Framework Description 39
3.4.1 Exploring Proximity Hints 41
3.4.2 Characteristic-rich Encoder 42
3.4.3 Difficulty-controllable Decoder 43
3.4.4 Training and Inference 44
3.5 Experiments 44
3.5.1 Experimental Settings 44
3.5.2 Difficulty Control Results 46
3.5.3 Manual Evaluation 49
3.5.4 Automatic Evaluation of Question Quality 51
3.5.5 Case Study 52
3.6 Summary 54

4 Distractor Generation for Multiple Choice Questions 55
4.1 Introduction 55
4.2 Framework Description 60
4.2.1 Task Definition 60
4.2.2 Framework Overview 60

4.2.3 Hierarchical Encoder 62
4.2.4 Static Attention Mechanism 63
4.2.5 Distractor Decoder 65
4.2.6 Training and Inference 68
4.3 Experimental Settings 68
4.3.1 Dataset 69
4.3.2 Implementation Details 70
4.3.3 Baselines and Ablations 71
4.4 Results and Analysis 73
4.4.1 Automatic Evaluation 73
4.4.2 Human Evaluation 74
4.4.3 Case Study 75
4.5 Summary 78

5 Conversational Question Generation 79
5.1 Introduction 80
5.2 Problem Setting 84
5.3 Model Description 84
5.3.1 Multi-Source Encoder 86
5.3.2 Decoder with Attention & Copy 87
5.3.3 Coreference Alignment 88
5.3.4 Conversation Flow Modeling 89
5.3.5 Joint Training 91
5.4 Experiments 91
5.4.1 Dataset Preparation 91
5.4.2 Implementation Details 92

5.4.3 Baselines and Ablations 93
5.5 Results and Analysis 94
5.5.1 Main Results 94
5.5.2 Coreference Alignment Analysis 95
5.5.3 Conversation Flow Modeling Analysis . . . 97
5.5.4 Human Evaluation 99
5.6 Summary 100

6 Explicit Memory Tracker for Conversational Machine Reading 101
6.1 Introduction 102
6.2 Method 105
6.2.1 Encoding 107
6.2.2 Explicit Memory Tracking 108
6.2.3 Decision Making 109
6.2.4 Follow-up Question Generation 111
6.3 Experiments 113
6.3.1 Experimental Setup 113
6.3.2 Results 116
6.3.3 Ablation Study 118
6.3.4 Interpretability 122
6.3.5 Error Analysis 123
6.4 Summary 124

7 Discourse-Aware Entailment Reasoning Network 125
7.1 Introduction 125
7.2 Discern Model 128

7.2.1 Rule Segmentation 129
7.2.2 Decision Making via Entailment Reasoning 131
7.2.3 Follow-up Question Generation 134
7.3 Experiments 135
7.3.1 Experimental Setup 135
7.3.2 Results 137
7.3.3 Ablation Study 139
7.3.4 Analysis of Logical Structure of Rules . . . 141
7.3.5 How Far Has the Problem Been Solved? . 142
7.4 Summary 145

8 Conclusion and Future Work 146
8.1 Conclusion 146
8.2 Future Work 147

Publications during Ph.D. Study 151

Bibliography 154

List of Figures

1.1 Sample question-answer pairs for knowledge assessment 3
1.2 Sample question-answer pairs for information acquisition 4

2.1 Taxonomy of existing question asking and answering tasks, methods, and our contributions. . . 13
2.2 An example SQuAD question. The answer ("0.3 °C") is highlighted. 22
2.3 Conversational Question Generation 23
2.4 Example of question asking and answering in conversational machine reading 27

3.1 Example easy and hard questions from SQuAD . 34
3.2 Overview of our DQG framework 40
3.3 Example questions generated by our model 53

4.1 Sample multiple choice question along with the corresponding article 57
4.2 A overview of our model that jointly utilizes static and dynamic attentions 61
4.3 Sample generated distractors 76

5.1 An example for conversational question generation 80
5.2 Passage chunks of interest for each turn chunks . 81
5.3 The framework of our proposed model 85
5.4 Examples for the coreference alignment model . . 96
5.5 The transition of passage attention distribution between turns 98

6.1 Example of Conversational Machine Reading tasks 103
6.2 Our framework: explicit memory tracker with coarse-to-fine reasoning 106
6.3 Predicted decisions and generated questions . . . 121

7.1 An example dialog from the ShARC dataset . . . 126
7.2 The overall diagram of our proposed DISCERN . . 130
7.3 Types of scenario interpretation errors based on 100 samples 144

8.1 An example for open-retrieval conversational machine reading 148
8.2 An example for answering ambiguous questions . 149

List of Tables

3.1 Distance statistics for non-stop words 41

3.2 The statistics of our DQG dataset 45

3.3 Difficulty of the generated questions 46

3.4 The results of controlling difficulty 48

3.5 Human evaluation results for generated questions 49

3.6 Automatic evaluation for question quality 51

4.1 The statistics of our distractor dataset 69

4.2 Automatic evaluation results on all systems by BLEU and ROUGE 72

4.3 Human evaluation results 74

5.1 Main results of baselines and our models 94

5.2 Evaluation results on the coreference test set . . . 95

5.3 Manual evaluation results 100

6.1 Performance on the test set of ShARC end-to-end task 116

6.2 Class-wise decision prediction accuracy 116

6.3 Performance on Oracle Question Generation Task 117

6.4 Ablation Study of EMT 119

7.1 Performance on the test set of ShARC end-to-end task 137
7.2 Class-wise decision prediction accuracy 138
7.3 Oracle question generation performance 139
7.4 Ablation study of DISCERN for decision making . 140
7.5 Decision prediction accuracy categorized by logical types of rules 141
7.6 Decision making and entailment prediction results over different subsets of the ShARC 143

Chapter 1

Introduction

1.1 Motivation

The capacity to use complex language to communicate and maintain our social world has been a trademark of humanity. Among different goals of language usage, questions and answers play significant roles in our day-to-day communications. We *ask* questions to explore unknown information from our side, and we *answer* questions to fill the information gap of others. As core skills to pass the Turing test, teaching machines to ask and answer questions has been studied for decades in Natural Language Processing (NLP) and Artificial Intelligence (AI) [75, 57, 4, 17]. Successful attempts bring us intelligent personal assistants such as Apple Siri, Microsoft Cortana, and Amazon Alexa. Moreover, asking and answering questions are becoming an essential part of existing major search engines such as Google conversational search and Alexa shopping in Amazon.

In natural language processing, question answering (QA) is defined as finding answers to natural language questions using certain knowledge sources. The knowledge source can be either structured such as a knowledge base, or unstructed such as a collection of natural language documents. Early attempts use some rule-based methods towards some specific domains, e.g., the LUNAR system developed in 1971 was able to answer 90% of the questions about the geological analysis of rocks returned by the Apollo moon missions. Then techniques evolve to statistical machine learning methods to answer questions in a probabilistic manner. In 2011, the IBM Watson question answering system competed in two exhibition matches of Jeopardy! against human beings by a significant margin. More recently, deep neural networks with billions of parameters have led to huge breakthroughs, outperforming human performance on several question answering benchmarks.

On the other hand, question asking (or question generation[1]) is the reverse task of question answering. It aims to generate natural language questions using certain knowledge sources. Back to the late 1960s, SHRDLU was developed to simulate the operation of a robot in a toy world, and it offered the possibility of asking the robot questions about the state of the world using some rule-based methods. In recent years, proactively asking questions has been adopted in modern task-oriented chatbots in e-commerce, online banking, and government websites. There are two main reasons to enable machines to ask questions:

- **Knowledge Assessment**: Asking questions is an effective approach to assess the knowledge understanding of users towards a specific topic. As shown in Figure 1.1, the machine asks several questions about a Wikipedia passage "Super Bowl 50" to test the user's knowledge understanding of this

[1]We use "question asking" and "question generation" interchangeably throughout this thesis.

Passage: Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi's Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the ”golden anniversary” with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as ”Super Bowl L”), so that the logo could prominently feature the Arabic numerals 50.

Q1: Which NFL team represented the AFC at Super Bowl 50?
A1: Denver Broncos
Q2: Which NFL team represented the NFC at Super Bowl 50?
A2: Carolina Panthers
Q3: Where did Super Bowl 50 take place?
A3: Santa Clara, California
Q4: Which NFL team won Super Bowl 50?
A4: Denver Broncos

Figure 1.1: Sample question-answer pairs about Super Bowl 50. Questions are asked for the purpose of knowledge testing with applications like intelligent e-tutor systems.

passage. Automatic question generation can be applied in the education domain such as intelligence tutor systems. Moreover, manually labelling question-answering pairs is laborious and requires some domain knowledge. Generating question-answer pairs is an essential data augmentation approach for training question answering models.

- **Information Acquisition**: If the user's original question is ambiguous or vague, machines have to first ask clarification questions before answering the original question. In dialogue systems, machines need to ask questions to gather information from users. As exemplified in Figure 1.2,

Rule Text 1: SBA provides loans **to businesses - not individuals** - so the requirements of eligibility are based on aspects of the business, not the owners. All businesses that are considered for financing under SBA's 7(a) loan program must: **meet SBA size standards**, be **for-profit**, **not already have the internal resources (business or personal) to provide the financing**, and be able to demonstrate repayment.
Rule Text 2: You'll need a statement of National Insurance you've paid in the UK to get these benefits - unless you're claiming Winter Fuel Payments.
Rule Text 3: 7(a) loans are the most basic and most used type loan of the Small Business Administration's (SBA) business loan programs. It's name comes from section 7(a) of the Small Business Act, which authorizes the agency to provide business loans to **American small businesses**. The loan program is designed to assist **for-profit businesses** that are **not able to get other financing from other resources**.

User Scenario: I am a 34 year old man from the United States who owns their own business. We are an American small business.
User Question: Is the 7(a) loan program for me?
Follow-up Q$_1$: Are you a for-profit business?
Follow-up A$_1$: Yes.
Follow-up Q$_2$: Are you able to get financing from other resources?
Follow-up A$_2$: No.
Final Answer: Yes. (You can apply the loan.)

Figure 1.2: An example to show the machine answers the user question by searching for relevant rule texts, reading rule texts, interpreting the user scenario, and keeping asking follow-up questions to clarify the user's background until it concludes a final answer.

the user asks whether he is eligible for the loan program. Before answering the question, the machine must gather relevant information from the user to evaluate his eligibility. Hence, asking clarification follow-up questions is helpful in conversation question answering.

As we can see, both *asking* and *answering* questions are essential to intelligent systems. In this thesis, we study question answering and asking in the reading comprehension scenario:

the knowledge source comes from a passage of text called a reading comprehension passage. Similar to how we evaluate a student's understanding of the reading comprehension text, we teach machines to ask and answer reading comprehension questions to evaluate its language understanding and its degree of AI. Under different goals of *knowledge assessment* and *information acquisition*, the contributions of this thesis consist of two parts:

- **Knowledge Assessment**: Question Asking in Reading Comprehension
  Given a passage of text, we teach machines to ask reading comprehension questions like "Which NFL team represented the NFC at Super Bowl 50?" to test human's understanding of the passage. In particular, we focus on 1) how to ask questions with different levels of difficulty, 2) how to generate distractors in multiple-choice questions, and 3) how to ask questions in a conversation to enhance the interactiveness and persistence of the knowledge testing process.

- **Information Acquisition**: Question Asking and Answering in Conversational Machine Reading
  Many questions cannot be answered directly in our day-to-day communications because we do not have enough background knowledge to give a concrete answer. Instead, we first ask questions to make clarifications or gather further information. In the second part of this thesis, we focus on teaching machines to ask questions when the original question requires additional information before responding.

1.2 Contributions

As previously mentioned, this dissertation will be mainly focused on question asking and answering for knowledge assessment and information acquisition. In the first part of this dissertation, we contribute to question asking from the knowledge assessment perspective. Our contributions can make the generated question difficulty controllable, make the generated options in multiple-choice questions more distracting, and make the question asking process more conversational:

- **Difficulty Controllable Question Generation**
  We investigate the difficulty levels of questions in reading comprehension datasets and propose a new question generation setting, named **D**ifficulty-controllable **Q**uestion **G**eneration (**DQG**). Taking as input a sentence in the reading comprehension paragraph and some of its text fragments (i.e., answers) that we want to ask questions about, a DQG method needs to generate questions under the control of specified difficulty labels—the output questions should satisfy the specified difficulty as much as possible. To solve this task, we propose an end-to-end framework to generate questions of designated difficulty levels by exploring a few important intuitions. Specifically, we explore a few intuitions: (i) In the input sentences, the nearer a word is to the answer fragment, the more likely it is used in the question; (ii) The easier a question is, the nearer its words are to the answer fragment in the sentence; (iii) Performing difficulty control could be regarded as a problem of sentence generation towards a specified attribute or style, namely difficulty level. For evaluation, we prepared the first

dataset of reading comprehension questions with difficulty labels. The results show that the question generated by our framework not only have better quality under the metrics like BLEU, but also comply with the specified difficulty labels.

- **Distractor Generation for Multiple Choice Questions**
  We investigate the task of distractor generation for multiple-choice reading comprehension questions from examinations. In contrast to all previous works, we do not aim at preparing words or short phrases distractors. Instead, we endeavor to generate longer and semantic-rich distractors which are closer to distractors in real reading comprehension from examinations. Taking a reading comprehension article, a pair of question and its correct option as input, our goal is to generate several distractors which are somehow related to the answer, consistent with the semantic context of the question and have some trace in the article. We propose a hierarchical encoder-decoder framework with static and dynamic attention mechanisms to tackle this task. Specifically, the dynamic attention can combine sentence-level and word-level attention varying at each recurrent time step to generate a more readable sequence. The static attention modulates the dynamic attention not to focus on question irrelevant sentences or sentences which contribute to the correct option. Our proposed framework outperforms several strong baselines on the first prepared distractor generation dataset of real reading comprehension questions. For human evaluation, compared with those distractors generated by baselines,

our generated distractors are more functional to confuse the annotators.

- **Conversational Question Generation**
  We study the problem of generating interconnected questions in question-answering style conversations. Compared with previous works which generate questions based on a single sentence (or paragraph), this setting is different in two major aspects: (1) Questions are highly conversational. Almost half of them refer back to conversation history using coreferences. (2) In a coherent conversation, questions have smooth transitions between turns. We propose an end-to-end neural model with coreference alignment and conversation flow modeling. The coreference alignment modeling explicitly aligns coreferent mentions in conversation history with corresponding pronominal references in generated questions, making generated questions interconnected to conversation history. The conversation flow modeling builds a coherent conversation by starting questioning the first few sentences in a text passage and smoothly shifting the focus to later parts. Extensive experiments show that our system outperforms several baselines and can generate highly conversational questions.

In the second part of this thesis, we focus on asking questions for information acquisition. One typical scenario is conversational machine reading, in which machines need to answer some high-level questions by asking some clarification follow-up questions first. We propose two complementary approaches to this problem:

- **Explicit Memory Tracker with Course-to-Fine Rea-**

soning
The goal of conversational machine reading is to answer user questions given a knowledge base text which may require asking clarification questions. Existing approaches are limited in their decision-making due to struggles in extracting question-related rules and reasoning about them. We present a new framework of conversational machine reading that comprises a novel **E**xplicit **M**emory **T**racker (EMT) to track whether conditions listed in the rule text have already been satisfied to make a decision. Moreover, our framework generates clarification questions by adopting a coarse-to-fine reasoning strategy, utilizing sentence-level entailment scores to weight token-level distributions. On the ShARC benchmark (blind, held-out) test set, EMT achieves new state-of-the-art results of 74.6% micro-averaged decision accuracy and 49.5 BLEU4. We also show that EMT is more interpretable by visualizing the entailment-oriented reasoning process as the conversation flows.

- **Discourse-Aware Entailment Reasoning Network**
  Document interpretation and dialog understanding are the two major challenges for conversational machine reading. In this chapter, we propose DISCERN, a discourse-aware entailment reasoning network, to strengthen the connection and enhance the understanding for both document and dialog. Specifically, we split the document into clause-like elementary discourse units (EDU) using a pre-trained discourse segmentation model. We train our model in a weakly supervised manner to predict whether the user feedback in a conversation entails each EDU. Based on

the learned EDU and entailment representations, we either reply to the user our final decision "yes/no/irrelevant" of the initial question or generate a follow-up question to inquiry more information. Our experiments on the ShARC benchmark (blind, held-out test set) show that DISCERN achieves state-of-the-art results of 78.3% macro-averaged accuracy on decision making and 64.0 BLEU1 on follow-up question generation.

1.3 Thesis Outline

We focus on teaching machines to ask and answer questions in this thesis. The remaining parts of this thesis can be divided into two main parts: 1) Knowledge Assessment in Reading Comprehension; 2) Information Acquisition in Conversational Machine Reading.

In the first part of this thesis, we focus on generating reading comprehension questions to test the knowledge understanding of humans towards a passage of text. Specifically, we investigate the difficulty levels of questions and propose a new question generation setting, named **D**ifficulty-controllable **Q**uestion **G**eneration (**DQG**) in Chapter 3. In Chapter 4, we investigate the task of distractor generation for multiple-choice questions from real examinations. After gaining the capability of generating questions more difficulty controllable and more distracting, we focus on making the question-answering process more interactiveness and persistence: we study the problem of conversational question generation in Chapter 5.

For the second part of this thesis, we start from the information acquisition perspective of our day-to-day communications. Specifically, we teach machines to answer high-level questions by firstly asking questions to gather information and answering the high-level question until enough clarifications have been made. Chapter 6 presents a new framework of conversational machine reading that comprises a novel **E**xplicit **M**emory **T**racker (EMT) to track whether conditions listed in the rule text have already been satisfied to make a decision. Chapter 7 proposes a discourse-aware entailment reasoning network to strengthen the connection and enhance the understanding for both document and dialog.

Finally, we conclude in Chapter 8 and list several potential future directions that deserve further exploration.

□ **End of chapter.**

Chapter 2

Background Review

In this chapter, we review background knowledge and related work of this thesis. We firstly review fundmantals of deep learning for natural language processing (NLP) in Section 2.1, including language models and representations, sequence-to-sequence models, and evaluation metrics. In Section 2.2, we present a literature review in Question Generation (QG). We examine the corpora, methodologies, and show how our contributions are related to these existing works. In Section 2.3, we survey machine reading comprehension and elaborate how question asking (generation) can help in question answering. Finally, we briefly introduce other tasks in question answering including Knowledge Base Question Answering (KBQA) and Community Question Answering (CQA) in Section 2.4. The taxonomy of question answering and where our contribution exists is shown in Figure 2.1.

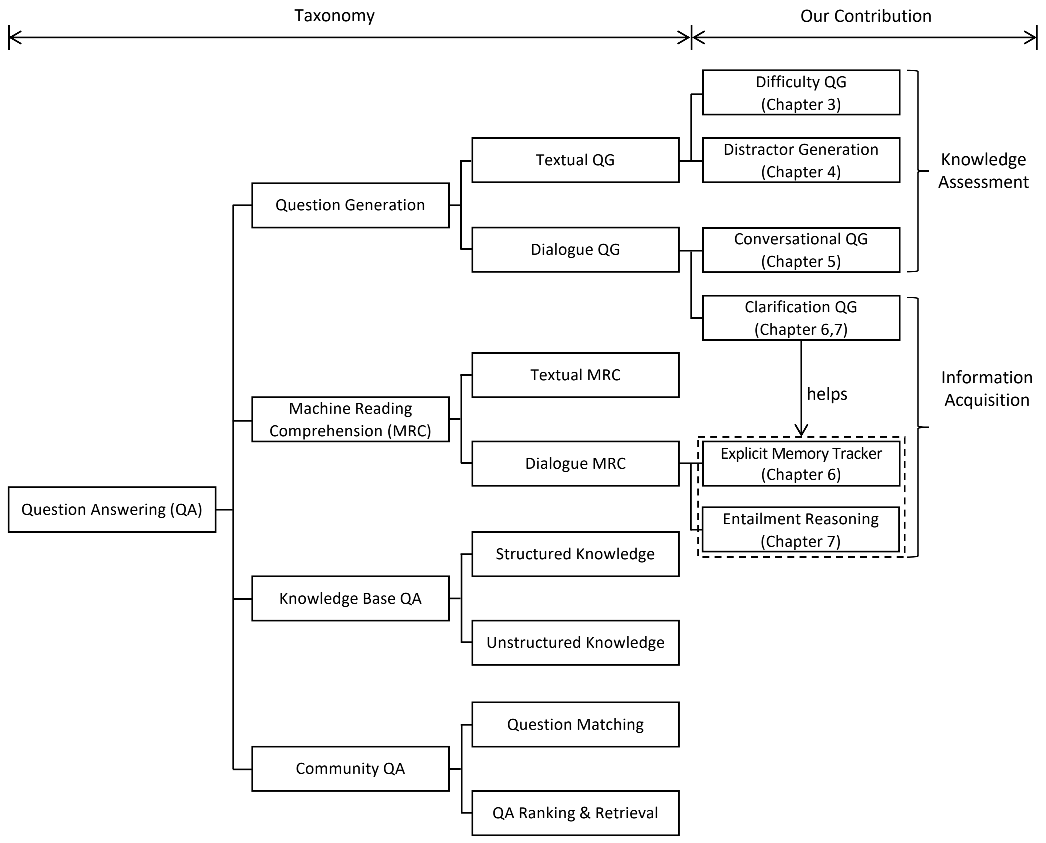


Figure 2.1: Taxonomy of existing question asking and answering tasks, methods, and our contributions.

2.1 Deep Learning Basics for NLP

With rapid progress in natural language processing, deep learning has been dominant in most NLP tasks and even outperformed human performance in some tasks. In this section, we will briefly introduce the major breakthrough of deep learning in natural language processing. The background knowledge described in this section is fundamental to our contributions in this thesis. We firstly introduce language models and language representations using recurrent neural network (RNN). We will then introduce the sequence-to-sequence learning framework for language generation tasks. Thirdly, we describe a new network architecture called Transformer which has been vastly used in the past three years as well as the language pretraining approaches based on the Transformer architecture. Finally, we talk about evaluation metrics in natural language processing tasks.

2.1.1 RNN and Language Models

Given a natural language sentence, language models compute the probability of the occurrence of the sentence in that particular word order. The probability of all m words in that particular sequence is denoted as $P(w_1, w_2, ..., w_m)$, where w_i is the i-th word in the sentence. The language model gives a score to represent the *goodness* of the sentence. For example, *He is walking home after school* will receive a higher score than *He is home walking after school* because the former sentence is grammatically correct. In existing machine translation and dialogue systems, the system will predict several plausible outputs. Then language model can give a *goodness* score for every prediction and the system will take the best one. Moreover, language models can suggest the next words given the partial observed sentence by computing $P(w_i|w_1, ..., w_{i-1})$. This is vastly used in the decoding process of conditional language models.

In accordance with the appearance order of words in the sentence, $P(w_1, w_2, ..., w_m)$ can be factorized as the multiplication of the probability of all words, conditioned on the partial observed sequence:

$$P(w_1, w_2, ..., w_m) = \prod_{i=1}^{m} P(w_i|w_1, ..., w_{i-1}). \tag{2.1}$$

The condiditional probability $P(w_i|w_1, ..., w_{i-1})$ can be modeled by Recurrent Neutal Network (RNN). Firstly, each word w in the sentence is mapped to its distributed word representation $\mathbf{w} \in \mathcal{R}^d$, which is a vectorized representation learned in training. Given a sentence with m words, the RNN are operated in m times to encode the seuqence, sequentially taking one word at a time. At time step t, RNN takes two inputs: the output of previous step $\mathbf{h}_{t-1}$ (called hidden state) and the vectorized representation of word $\mathbf{w}_t$. Then it performs a linear matrix operation on its inputs followed by a non-linear operation to predict the probability of the next word $P(w_{t+1}|w_1, ..., w_t)$:

$$\mathbf{h}_t = sigmoid(\mathbf{W}_1\mathbf{h}_{t-1} + \mathbf{W}_2\mathbf{w}_t), \tag{2.2}$$

$$P(w_{t+1}|w_1, ..., w_t) = softmax(\mathbf{W}_3\mathbf{h}_t), \tag{2.3}$$

where $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$ are learnable parameters.

Recurrent neural network works perfectly to encode all the previous tokens for language modeling, which is infeasible for non-neural-network approaches. However, RNN has gradient vanishing issue that limits its modeling capability to short sentences. Long Short Term Memory (LSTM) [27] units and Gated Recurrent Units (GRU) [7] are proposed to address this issue.

2.1.2 Sequence-to-Sequence Models

Language models can either give a *goodness* score for a sentence or predict the next word based on the partial observed sentence. However, many NLP tasks require mapping one sequence of words to another sequence of words. For example, machine translation systems translate a sentence from the source language to a sentence in the target language; summarization systems summarize a long passage of text to a short but concise sentence. Dialogue systems generate a response conditioned on the received utterance. All these tasks can be taken as a sequence-to-sequence learning problem. Conditional language models are proposed to achieve the mapping between two sequences.

The sequence-to-sequence model [82] is proposed to firstly used in neural machine translation. Then it is applied to all NLP tasks for conditional language modeling. In essence, sequence-to-sequence model is made up with two recurrent neural networks (RNN). Given a source sentence of m words $w^s_1, ..., w^s_m$, the *encoder* RNN encode the input sentence to its vectorized representation $\mathbf{h}^s_1, ..., \mathbf{h}^s_m$:

$$\mathbf{h}^s_t = \mathrm{RNN}_1(\mathbf{h}^s_{t\text{-}1}, \mathbf{w}^s_t). \tag{2.4}$$

Given the encoded input sequence $\mathbf{h}^s_1, ..., \mathbf{h}^s_m$, the second *decoder* RNN generates the output sentence step-by-step. At time step t, it takes the previously predicted word w^o_{t-1} (its vector $\mathbf{w}^o_{t-1}$) and the hidden state of the decoder RNN $\mathbf{h}^o_{t-1}$ to compute the hidden state for the current step,

$$\mathbf{h}^o_t = \mathrm{RNN}_2(\mathbf{h}^o_{t-1}, \mathbf{w}^o_{t-1}). \tag{2.5}$$

In addition, the decoder adopts an attention mechanism to derive a context vector that captures relevant input-side information to help predict the current target word:

$$\alpha_{i,t} = \frac{\exp(\mathbf{h}^o_t \mathbf{W}_1 \mathbf{h}^s_i)}{\sum_j \exp(\mathbf{h}^o_t \mathbf{W}_1 \mathbf{h}^s_j)}, \tag{2.6}$$

$$\mathbf{c}_t = \sum_i \alpha_{i,t} \mathbf{h}^s_i. \tag{2.7}$$

Then the decoder will predict the distribution of the next word based on the context vector $\mathbf{c}_t$ and the decoder hidden state $\mathbf{h}_t^o$:

$$P(w_t^o|w_{1,\dots,m}^s, w_{1,\dots,t\text{-}1}^o) = softmax(\mathbf{W}_2 tanh(\mathbf{W}_3[\mathbf{h}_t; \mathbf{c}_t])). \quad (2.8)$$

2.1.3 Transformer and Pretraining

Transformer [88] is a prominent deep learning model which was initially proposed for neural machine translation. It outperforms all RNN-based models on neural machine translation and is adopted to various natural language processing tasks. Later works show that Transformer-based pretrained language models can boost the performance of many natural language understanding and generation tasks [11, 38].

The vanilla Transformer model is a sequence-to-sequence model that includes an encoder part and a decoder part. Both encoder and decoder contains L layers of Transformer blocks. The Transformer block contains a multi-head self-attention module and a position-wise feed-forward network module. The multi-head self-attention module takes the query $\mathbf{q} \in \mathcal{R}^d$, keys $\mathbf{K} \in \mathcal{R}^{m \times d}$ and values $\mathbf{V} \in \mathcal{R}^{m \times d}$ (encoded from the source sequence) to perform the the scaled dot-product attention:

$$\text{Attention}(\mathbf{q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{q}\mathbf{K}^\top}{\sqrt{d}})\mathbf{V}, \quad (2.9)$$

where m denotes the input sequence length, d denotes the dimension of vectors. Instead of applying the attention a single time, Transformer applies multi-head attention to practice it multiple times. Each attention head will learn part of the semantic for the input sentence. Then the position-wise feed-forward network concatenates all attention vectors together and then pass to a linear layer to get the final representation. The remaining part of Transformer follows the same encoding and decoding process with RNN-based sequence-to-sequence modeling. Besides sequence-to-sequence modeling, the Transformer encoder and decoder can be separately used for language understanding tasks and language modeling tasks.

Different from recurrent neural network that inherently introduces the inductive bias locally, Transformer has no assumption over the structure of the input sequence in its self-attention. Because of this, Transformer can capture dependencies in the input sequence at any range. This capability makes Transformer learn universal language representation from large-scale pretraining. The inductive bias learned from large corpora is extremely powerful to various downstream tasks [11]. During the pretraining stage, Transformer is trained on some self-supervised loss. For example, one kind of self-supervised loss named masked language modeling loss is designed to make the model predict a masked word given its context. After pretraining, the pretrained model can be finetuned with a small-scale downstream data. There are three major styles for using Transformer in pretraining:

- Encoder Only: Pretraining with Transformer is firstly successful by adopting its encoder part only in BERT [11]. BERT has a masked language modeling loss and a next sentence prediction loss to train the Transformer encoder. Then it can be finetuned for many language understanding tasks, such as text classification, reading comprehension, and textual entailment.

- Decoder Only: The Generative Pretrained Transformer

(GPT) [65] is the first model trained using Transformer decoder. The trained Transformer decoder can be used for language modeling related tasks.

- Encoder-Decoder: To make large-scale pretraining work for sequence-to-sequence tasks, researchers propose to to train the encoder-decoder Transformer model directly. BART [38] extends the denoising objective of BERT to encoder-decoder architecture. In addition to performing conditional language generation tasks, the encoder-decoder pretrained model can also perform language understanding tasks.

2.1.4 Evaluation Metrics

In natural language processing, many tasks have unique evaluation metrics to assess the language understanding/generation performance. We categorize them into three primary evaluation metrics: classification tasks, span extraction tasks, and language generation tasks.

Classification Tasks. Many language understanding tasks can be formulated as a classification problem. For example, sentiment analysis requires taking a sentence and predicting several levels of sentiment ranging from positive to negative. For these classification tasks, classical evaluation metrics are used such as precision, recall, F-score, and accuracy.

Span Extraction Tasks. In natural language processing, sometimes the output is a text span inside the input sentence. For example, keyphrase extraction requires predicting several spans inside the sentence as keyphrase; reading comprehension requires predicting a span within the text as the answer to the input question. In this case, span extraction tasks have their own evaluation metric: Exact Match (EM) and F1. EM treats the prediction correct if and only if the predicted span is exactly the same as the ground truth span, while F1 takes partial match into account by computing the F-score between the predicted phrase (multiple words) and the gold phrase. Then we take the average performance of all predicted samples as the final EM/F1 score.

Generation Tasks. The outputs of natural language generation tasks are sentences, making these tasks hard to evaluate because we need to measure the semantic similarity between the predicted and gold sentences. Nevertheless, there are some evaluation metrics developed for machine translation and text summarization:

- BLEU: BiLingual Evaluation Understudy [59] is an algorithm for evaluating the machine translation systems. It compares the n-grams accuracy between the machine-translated sentence with several reference sentences (gold sentences) with a penalty for predictions that are shorter than the references. BLEU is the first metric to claim a high correlation with human judgments of quality.
- ROUGE: Recall-Oriented Understudy for Gisting Evaluation [45] is a set of metrics for evaluating text summarization systems. Unlike BLEU, ROUGE focuses on the precision of the predicted sentence and takes recall into account since text summarization requires summarizing text within a predefined length.

The outputs for BLEU and ROUGE are always a number between 0 and 1. This value indicates how similar the candidate text is to the reference texts, with values closer to 1 representing more similar texts.

In addition to the automatic evaluation using BLEU and ROUGE, manual evaluation is necessary for other language generation tasks such as dialogue generation, question generation, and story generation. Since people can express the same meaning in a completely different utterance, the matching-based metrics are likely to fail to evaluate the correctness of predictions in the open-ended language generation tasks.

2.2 Question Generation

Question Generation (QG) aims to generate natural and human-like questions from a range of data sources, such as image [58], knowledge base [73, 78], and free text [15]. Besides constructing SQuAD-like dataset [66] for data augmentation in reading comprehension, QG is also helpful for the intelligent tutor system: The instructor can actively ask the learner questions according to reading comprehension materials [25] or particular knowledge [10]. For example, Figure 2.2 gives one question from SQuAD, the goal of question generation is to generate such questions.

2.2.1 Textual QG

QG for reading comprehension is a challenging task because the generation should follow the syntactic structure of questions and ask questions to the point, i.e., having a specified aspect as its answer. Some template-based approaches [87, 53, 47, 2, 25] were

Sentence: The daily mean temperature in January, the area's coldest month, is 32.6 °F (0.3 °C); however, temperatures usually drop to 10 °F (-12 °C) several times per winter and reach 50 °F (10 °C) several days each winter month.
Reference Question: What is New York City 's daily January mean temperature in degrees celsius ?

Figure 2.2: An example SQuAD question. The answer ("0.3 °C") is highlighted.

proposed initially. They manually design some question templates and transform the declarative sentences to interrogative questions [53, 34, 47, 25]. These Rule-based approaches need extensive human labor to design question templates and usually can only ask annotators to evaluate the generated questions.

With the rise of data-driven learning approach and sequence to sequence (seq2seq) framework [82], some researchers formulated QG as a seq2seq problem [15]: The question is regarded as the decoding target from the encoded information of its corresponding input sentence. However, different from existing seq2seq learning tasks such as machine translation and summarization which could be loosely regarded as learning a one-to-one mapping, for question generation, various aspects of the given descriptive sentence can be asked. Hence, the generated questions could be significantly different. Several recent works tried to tackle this problem by incorporating the answer information to indicate what to ask about, which helps the models generate more accurate questions [29, 76, 108]. Some researchers focus on how to utilize the answer information better to generate questions to the point [108, 81] and how to effectively use the contexts in paragraphs to generate questions that cover context beyond a single sentence [105, 14].

Passage: Incumbent Democratic President Bill Clinton was ineligible to serve a third term due to term limitations in the 22nd Amendment of the Constitution, and Vice President Gore was able to secure the Democratic nomination with relative ease. Bush was seen as the early favorite for the Republican nomination and, despite a contentious primary battle with Senator John McCain and other candidates, secured the nomination by Super Tuesday. Bush chose ...

Q_1:	What political party is Clinton a member of?	A_1:	Democratic
Q_2:	What was he ineligible to serve?	A_2:	third term
Q_3:	Why?	A_3:	term limitations
Q_4:	Based on what amendment?	A_4:	22nd
Q_5:	Of what document?	A_5:	Constitution
Q_6:	Who was his vice president?	A_6:	Gore
Q_7:	Who was the early Republican favorite for the nomination?	A_7:	Bush
Q_8:	Who was the primary battle with?	A_8:	John McCain
Q_9:	What is his title?	A_9:	Senator
Q_{10}:	When did Bush secure the nomination by?	A_{10}:	Tuesday

Figure 2.3: An example for conversational question generation from a conversational question answering dataset CoQA [67]. Each turn contains a question Q_i and an answer A_i.

2.2.2 Dialogue QG

Different from formalizing question generation for knowledge testing as a standalone interaction [104, 76], it is a more natural way for human beings to test knowledge or seek information through conversations involving a series of interconnected questions [67]. As shown in Figure 2.3, the machine starts a conversation with the student, and asks a series of conversational questions to test his understanding of the textual passage. Different from the standalone interaction in Figure 2.2, questions in Figure 2.3 are highly conversational. The inability for virtual assistants to ask questions based on previous discussions often leads to better user experiences. Starting from this motivation, some researchers recently investigate question generation in dialogue systems. [41] show that asking questions through interactions can receive useful feedbacks to reach the correct answer. [93] consider asking questions in open-domain conversational systems with typed decoders to enhance the interactiveness and persistence of conversations.

Our Contributions. Under the same goal for knowledge assessment, our contributions are following:

- Textual QG: First, we propose to model the difficulty of questions in question generation. Then we propose to generate questions under the predefined difficulty. Second, we propose a new framework to generate distractors (wrong options) for multiple choice questions in reading comprehension.
- Dialogue QG: We propose to generate interconnected questions in a question-answering style conversation. The new dialogue-based interaction style can enhance the interactiveness and persistence of knowledge assessment.

2.3 Machine Reading Comprehension

Machine Reading Comprehension (MRC) is a long-standing goal in natural language understanding with various applications in question answering and dialogue systems. The task formulation of machine reading comprehension is as follows: Given a textual passage P and a question Q over the content of the passage, a MRC system predicts an answer A which could be a text span within the textual passage P. Besides the span-based answers, MRC can also be formulated to predict a masked entity within the passage (cloze-style MRC), or select a option within multiple provided choices (multi-choice MRC), or generate free form answers that may or may not appear in the passage (generative MRC). However, span-extraction style MRC is the most popular task because of the well-defined target and convenience for evaluation.

2.3.1 Textual MRC

Textual Machine Reading Comprehension keeps the simplest setting. SQuAD [66] is a representative task in textual MRC which includes 100,000 question-passage pairs. The passages are collected from Wikipedia while the questions are generated by crowd workers. The answer for the SQuAD task is in the format of text span inside the passage.

There are two major architecture for textual machine reading comprehension task: traditional RNN-based matching model [72] and pretrained transformer-based model [11, 49]. Traditional RNN-based matching models adopt RNN as a feature encoder to encode the question and passage separately. Then a question-passage attention mechanism is designed for matching interaction. Finally, start and end indices are predicted for the answer span. Pretrained transformer-based models rely heavily on the language representations learned in the pretraining stage. They concatenate the passage with the question into a sequence and feed it for pretrained models. The multi-head self-attention of transformers can replace RNN encoding and complicated question-passage matching. A simple linear layer is added on top of pretrained models to predict the start and end position for the answer span.

2.3.2 Dialogue MRC

In conversational machine reading (CMR) comprehension, machines can take the initiative to ask users questions that help to solve their problems, instead of jumping into a conclusion hurriedly [69]. In this case, machines need to understand the knowledge base (KB) text, evaluate and keep track of the user scenario, ask clarification questions, and then make a final decision. This interactive behavior between users and machines has gained more attention recently because in practice users are unaware of the KB text, thus they cannot provide all the information needed in a single turn.

For instance, consider the example in Figure 2.4 taken from the ShARC dataset for CMR [69]. A user posts her scenario and asks a question on whether her employer can take money from her final pay. Since she does not know the relevant rule text, the provided scenario and the initial question(s) from her are often too underspecified for a machine to make a certain decision. Therefore, a machine has to read the rule text and ask a series of clarification questions until it can conclude the conversation with a certain answer.

Most existing approaches [107, 89] formalize the CMR problem into two sub-tasks. The first is to make a decision among `Yes`, `No`, `Irrelevant`, and `Inquire` at each dialog turn given a rule text, a user scenario, an initial question and the current dialog history. If one of `Yes`, `No`, or `Irrelevant` is selected, it implies that a final decision (`Yes`/`No`) can be made in response to the user's initial question, or stating the user's initial question is unanswerable (`Irrelevant`) according to the rule text. If the decision at the current turn is `Inquire`, it will

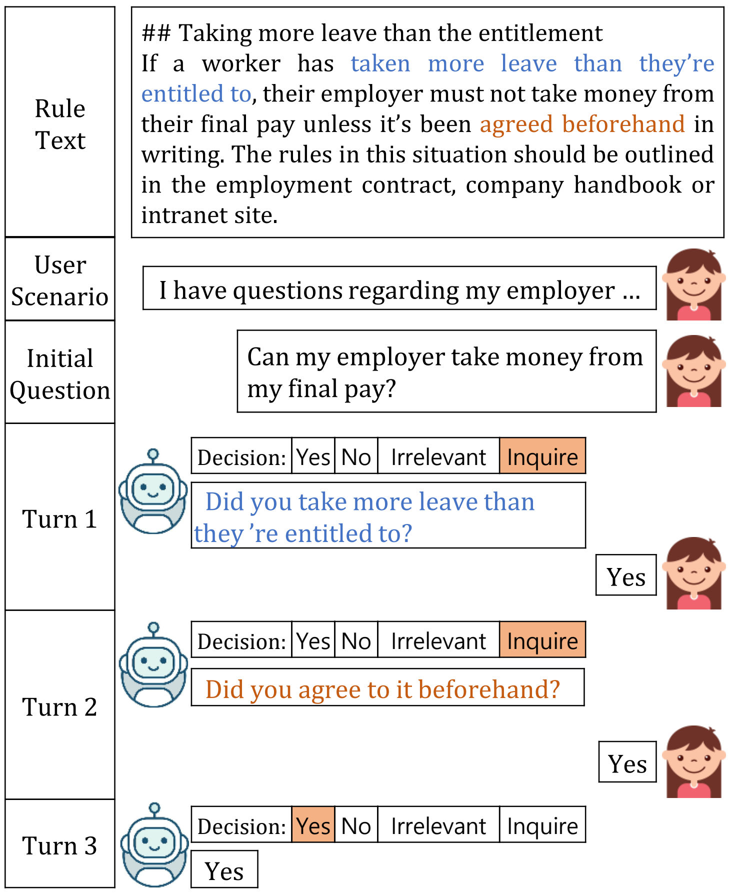

Rule Text
## Taking more leave than the entitlement
If a worker has taken more leave than they're entitled to, their employer must not take money from their final pay unless it's been agreed beforehand in writing. The rules in this situation should be outlined in the employment contract, company handbook or intranet site.
User Scenario
I have questions regarding my employer ...
Initial Question
Can my employer take money from my final pay?
Turn 1
Decision: Yes No Irrelevant Inquire
Did you take more leave than they 're entitled to?
Yes
Turn 2
Decision: Yes No Irrelevant Inquire
Did you agree to it beforehand?
Yes
Turn 3
Decision: Yes No Irrelevant Inquire
Yes

Figure 2.4: Example of Conversational Machine Reading tasks from the ShARC dataset [69]. At each turn, given the rule text, a user scenario, an initial user question, and previous interactions, a machine can give a certain final answer such as `Yes` or `No` to the initial question. If the machine cannot give a certain answer because of missing information from the user, it will ask a clarification question to fill in the information gap. Clarification questions and their corresponding rules are marked in the same colors.

then trigger the second task for follow-up question generation, which extracts an underspecified rule span from the rule text and generates a follow-up question accordingly. On the ShARC CMR challenge [69], [37] propose an end-to-end bidirectional sequence generation approach with mixed decision making and question generation stages. [69] split it into sub-tasks and combines hand-designed sub-models for decision classification, entailment and question generation. [107] propose to extract all possible rule text spans, assign each of them an entailment score, and edit the span with the highest score into a follow-up question. However, they do not use these entailment scores for decision making. [89] study patterns of the dataset and include additional embeddings from dialog history and user scenario as rule markers to help decision making.

Conversational Machine Reading [69] differs from conversational question answering [8, 67] and conversational question generation [19] in that 1) machines are required to formulate follow-up questions to fill the information gap, and 2) machines have to interpret a set of complex decision rules and make a question-related conclusion, instead of extracting the answer from the text. CMR can be viewed as a special type of task-oriented dialog systems [95, 106, 97] to help users achieve their goals. However, it does not rely on predefined slot and ontology information but natural language rules.

Our Contributions. We focus on the dialogue machine reading comprehension in this thesis. Different from previous works which aim to achieve better question answering performance in dialogue MRC, we aim to change the interaction style in dialogue MRC in which the machine can not only passively answer questions but also proactively ask clarification questions. The new interaction style can improve the overall MRC system performance. Specifically, our contributions are as follows:

- We propose an explicit memory tracker to track the acquired knowledge in the dialogue. The machine can ask questions accordingly when there is any knowledge underspecified.
- We propose to adopt discourse parsing to understand the passage and entailment reasoning to understand the dialogue. The enhanced understanding of both the passage and the text leads to better QA performance.

2.4 Other QA Tasks

2.4.1 Knowledge-Based Question Answering

Knowledge-Based Question Answering (KBQA) aims to answer questions with the help of knowledge bases. Unlike machine reading comprehension which has a passage for reference before answering the question, KBQA only provides the same knowledge source for answering all questions. Knowledge sources in KBQA can be categorized into structured knowledge base and unstructured knowledge base. Structured knowledge base is in the format of KB triple. For example, given the question "How tall is Yao Ming?", the model will find the KB triple ((Yao Ming, height, 2.26m) and take "2.26m" as the answer. Unstructured knowledge base is free text knowledge such as Wikipedia. For the same question on Yao Ming's height, the model needs to retrieve the key sentence "During his final season, he was the tallest active player in the NBA, at 2.29 m (7 ft 6 in).” in Wikipedia and predict the answer accordingly.

Approaches for KBQA can be divided into two classes: information retrieval-based and neural semantic parsing-based. Information-retrieval based methods [3, 24, 80] firstly retrieve a large set of candidate answers from the knowledge base according to the matching score between the question and answers. Then these candidate answers are reranked and sorted to select the final prediction. Some IR-based methods [36] have the iterative retrieve-read mechanism to perform multi-hop reasoning to handle complex questions. However, these methods are lack of interpretability and cannot tackle complex questions requiring constraint inference.

Different from IR-based methods that directly encode the question into a vectorized representation, semantic parsing-based methods aim to convert the natural language query into an executable language query. These methods [68, 102, 50, 36] usually convert the natural language question into logical forms such as query graphs and trees and further translate them into SPARQL queries. Recently, some researchers also investigate encoder-decoder based approaches [12, 98] to leverage trees or high-level programming languages to represent natural language questions.

2.4.2 Community Question Answering

Community Question Answering (CQA) is a dominaint approach for online information seeking, where users can ask and answer questions on web forums such as Yahoo! Answers[1], Stack

[1]`https://answers.yahoo.com/`

Overflow[2], Quora[3]. Usually, these forums have multiple domain experts to provide answers or opinions to user-posted questions. Different from machine reading comprehension and knowledge-based QA, community question answering is different in the following aspects:

- Question Type: Community question answering have multi-sentence questions while other QA tasks aim to answer single sentence factoid questions.
- Source of Answers: Community question answering has user-generated long and comprehensive answers while answers in other QA tasks are mostly extracted from the context passage or knowledge base.
- Meta Data: Different from the simple formulation of QA, community question answering evolves social computing research. One major characteristic of CQA is that users can upvote/downvote answers to select the best answer.

Because of the above difference, community question answering has a different research focus. For a reliable online QA forum, there are two major tasks in community question answering: 1) Question Matching, and 2) QA Ranking & Retrieval.

Question Matching. In an online forum, users repeatedly ask questions that share the same meanings. This is called the question duplication problem. Solving question duplication will lead to less redundancy for experts so that they do not

[2] `https://www.stackoverflow.com/`

[3] `https://www.quora.com/`

need to answer similar questions again and again. Moreover, it is easier for users to find existing answers when they post questions. The task is to match the new question to all questions in the database. Traditional methods like BM25 evaluate the similarity between questions by matching tokens. In the era of deep learning, some neural network-based methods [60] are developed. Questions are encoded via RNN or pretrained language models, and then the two question representations are concatenated, and a dense layer is applied to generate the prediction.

QA Ranking & Retrieval. Given a list of user-generated answers, it is essential to rank and retrieve the best response within them. Similar to the question matching task, QA ranking & retrieval can utilize convolutional neural network [16], recurrent neural network [85], or pretrained language models to encode the vectorized representation of the question and candidate answers. Then the questions and answer vectors are trained under the log-likelihood loss by predicting the answer is relevant to the question or not. Moreover, some researchers propose to model the token-level interaction between the question and candidate answers through attention mechanism [62]. The proposed approach receives better performance for answer relevance while sacrificing computation efficiency.

□ **End of chapter.**

Chapter 3

Difficulty Controllable Question Generation

In this chapter, we investigate the difficulty levels of questions in reading comprehension datasets such as SQuAD, and propose a new question generation setting, named **D**ifficulty-controllable **Q**uestion **G**eneration (**DQG**). We first give an introduction in Section 3.1. In Section 3.2, we define our proposed new task – difficulty controllable question generation. Section 3.3 describes our protocol to label the difficulty levels of questions. In Section 3.4, we introduce our proposed end-to-end framework to generate questions of designated difficulty levels by exploring a few important intuitions. Section 3.5 presents our experiment results. Finally, we give a summary of this chapter in Section 3.6.

3.1 Introduction

Question Generation (QG) aims to generate natural and human-like questions from a range of data sources, such as image [58], knowledge base [73, 78], and free text [15]. Besides for

S1: Oxygen is a chemical element with symbol O and atomic number 8.
Q1: (*Easy*) What is the atomic number of the element oxygen?
A1: 8

S2: It is a member of the chalcogen group on the periodic table and is a highly reactive nonmetal and oxidizing agent that readily forms compounds (notably oxides) with most elements.
Q2: (*Easy*) Of what group in the periodic table is oxygen a member?
A2: chalcogen

S3: The electric guitar is often emphasised, used with distortion and other effects, both as a rhythm instrument using repetitive riffs with a varying degree of complexity, and as a solo lead instrument.
Q3: (*Hard*) What instrument is usually at the center of a hard rock sound?
A3: The electric guitar

Figure 3.1: Example questions from SQuAD. The answers of Q1 and Q2 are facts described in the sentences, thus they are easy to answer. But it is not straightforward to answer Q3.

constructing SQuAD-like dataset [66], QG is also helpful for the intelligent tutor system: The instructor can actively ask the learner questions according to reading comprehension materials [25] or particular knowledge [10]. In this chapter, we focus on QG for reading comprehension text. For example, Figure 3.1 gives three questions from SQuAD. Our goal is to generate such questions.

QG for reading comprehension is a challenging task because the generation should not only follow the syntactic structure of questions, but also ask questions to the point, i.e., having a specified aspect as its answer. Some template-based approaches [87, 53, 47, 2, 25] were proposed initially, where well-designed rules and heavy human labor are required for declarative-to-interrogative sentence transformation. With the rise of data-driven learning approach and sequence to sequence (seq2seq) framework [82], some researchers formulated QG as a seq2seq problem [15]: The question is regarded as the decoding target from the encoded information of its corresponding input sentence. However, different from existing seq2seq learning tasks such as machine translation and summarization which could be loosely regarded as learning a one-to-one mapping, for question generation, different aspects of the given descriptive sentence can be asked, and hence the generated questions could be significantly different. Several recent works tried to tackle this problem by incorporating the answer information to indicate what to ask about, which helps the models generate more accurate questions [29, 76, 108]. In our work, we also focus on the answer-aware QG problem, which assumes the answer is given. Similar problems have been addressed in, e.g., [105, 81]

In this chapter, we investigate a new setting of QG, namely **D**ifficulty controllable **Q**uestion **G**eneration (**DQG**). In this setting, given a sentence in the reading comprehension paragraph, the text fragments (i.e., answers) that we want to ask questions about, and the specified difficulty levels, a framework needs to generate questions that are asked about the specified answers and satisfy the difficulty levels as much as possible. For example, given the sentence S3 and the answer “the electric guitar” in Figure 3.1, the system should be capable of asking both a hard question like Q3 and an easy one such as “What is often emphasised as a rhythm instrument?”. DQG has rich application scenarios. For instance, when instructors prepare learning materials for students, they may want to balance the numbers of hard questions and easy questions. Besides, the generated questions can be used to test how a QA system works for questions with diverse difficulty levels.

Generating questions with designated difficulty levels is a more challenging task. First, no existing large-scale QA dataset has difficulty labels for questions for training reliable neural network models. Second, for a single sentence and answer pair, we want to generate questions with diverse difficulty levels. However, the current datasets like SQuAD only have one given question for each sentence and answer pair. Finally, there is no metric to evaluate the difficulty of questions. To overcome the first issue, we prepare a dataset of reading comprehension questions with difficulty labels. Specifically, we design a method to automatically label SQuAD questions with multiple difficulty levels, and obtain 76K questions with difficulty labels.

To overcome the second issue, we propose a framework that can learn to generate questions complying with the specified difficulty levels by exploring the following intuitions. To answer a SQuAD question, one needs to locate a text fragment in the input sentence as its answer. Thus, if a question has more hints that can help locate the answer fragment, it would be easier to answer. For the examples in Figure 3.1, the hint "atomic number" in Q1 is very helpful, because, in the corresponding sentence, it is just next to the answer "8", while for Q3, the hint "instrument" is far from the answer "The electric guitar". The second intuition is inspired by the recent research on style-guided text generation, which incorporates a latent style representation (e.g., sentiment label or review rating score) as an input of the generator [74, 44]. Similarly, performing difficulty control can be regarded as a problem of sentence generation towards a specified attribute or style. On top of the typical seq2seq architecture, our framework has two tailor-made designs to explore the above intuitions: (1) Position embeddings are learned to capture the proximity hint of the answer in the input sentence; (2) Global difficulty variables are learned to control the overall "difficulty" of the questions. For the last issue, we propose to employ the existing reading comprehension (RC) systems to evaluate the difficulty of generated questions. Intuitively, questions which cannot be answered by RC systems are more difficult than these correctly answered ones.

In the quantitative evaluation, we compare our DQG model with state-of-the-art models and ablation baselines. The results show that our model not only generates questions of better quality under the metrics like BLEU and ROUGE, but also has the capability of generating questions complying with the specified difficulty labels. The manual evaluation finds that the language quality of our generated questions is better, and our model can indeed control the question difficulty. We plan to release the prepared dataset and the code of our model for further research.

3.2 Task Definition

In the DQG task, our goal is to generate SQuAD-like questions of diverse difficulty levels for a given sentence. Note that the answers of SQuAD questions are text spans in the input sentence, and they are significantly different from RACE questions [35] such as "What do you learn from the story?" Considering their different emphases, SQuAD questions are more suitable for our task, while the difficulty of RACE questions mostly comes from the understanding of the story but not from the way how the question is asked. Thereby, we assume that the answers for asking questions are given, and they appear as text fragments in the input sentences by following the paradigm of SQuAD.

We propose an end-to-end framework to handle DQG. Formally, let a denote the answer for asking question, and let s denote the sentence containing a from a reading comprehension paragraph. Given a, s, and a specified difficulty level d as input, the DQG task is to generate a question q which has a as its answer, and meanwhile should have d as its difficulty level.

3.3 The Protocol of Difficulty Labeling

SQuAD [66] is a reading comprehension dataset containing 100,000+ questions on Wikipedia articles. The answer of each question is a text fragment from the corresponding input passage. We employ SQuAD questions to prepare our experimental dataset.

The difficulty level is a subjective notion and can be addressed in many ways, e.g., syntax complexity, coreference resolution and elaboration [79]. To avoid the ambiguity of the "question difficulty" in this preliminary study, we design the following automatic labeling protocol and study the correlation between automatically labelled difficulty with human difficulty. We first define two difficulty levels, *Hard* and *Easy*, in this preliminary dataset for the sake of simplicity and practicality. We employ two RC systems, namely R-Net [92] [1] and BiDAF [72] [2], to automatically assess the difficulty of the questions . The *labeling protocol* is: A question would be labelled with *Easy* if both R-Net and BiDAF answer it correctly under the exact match metric, and labelled with *Hard* if both systems

[1] `https://github.com/HKUST-KnowComp/R-Net`

[2] `https://github.com/allenai/bi-att-flow`

fail to answer it. The remaining questions are eliminated for suppressing the ambiguity.

Note that we cannot directly employ the original data split of SQuAD to train a model of R-Net or BiDAF, and use the model to assess all questions. Such assessment does not make sense, because some questions (i.e., training and validation questions) are already shown to the model in the training. To avoid this problem, we re-split the original SQuAD questions into 9 splits and adopt a 9-fold strategy. To label every single split (the current split), 7 splits are used as the training data, and the last split is used as the validation data. Then the trained model is used to assess the difficulty of questions in the current split. This way guarantees that the model is never shown with the questions for automatic labeling. Finally, we obtain 44,723 easy questions and 31,332 hard questions.

To verify the reasonability of our labeling protocol, we evaluate its consistency with human being's judgment. We sample 100 *Easy* questions and 100 *Hard* questions, and hire 3 annotators to rate the difficulty level of all these questions on a 1-3 scale (3 for the most difficult). The result shows that average difficulty rating for the *Easy* questions is 1.90 while it is 2.52 for the *Hard* ones.

3.4 Framework Description

Given an input sentence $s = (w_1, w_2, ..., w_m)$, a text fragment a in s, and a difficulty level d, our task is to generated a question q, which is asked with s as its background information, takes a as its answer, and has d as its difficulty. The architecture

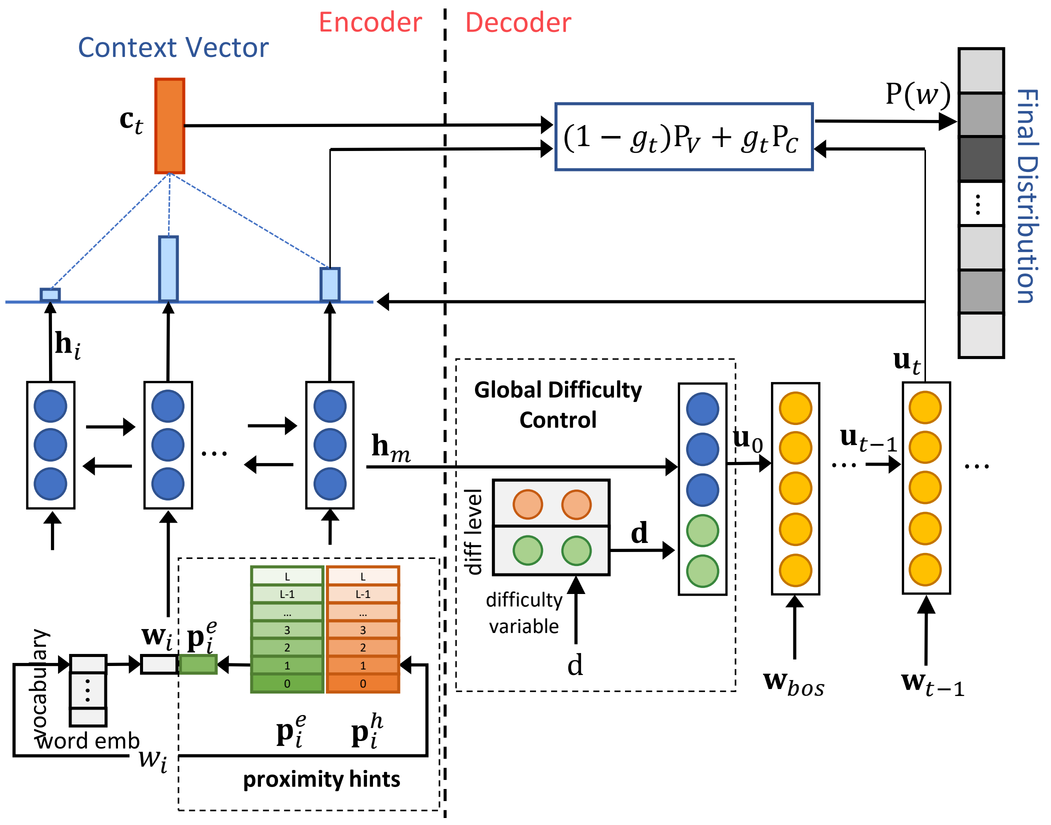


Figure 3.2: Overview of our DQG framework *(better viewed in color)*.

of our difficulty-controllable question generator is depicted in Figure 3.2. The encoder takes two types of inputs, namely, the word embeddings and the relative position embeddings (capturing the proximity hints) of sentence words (including the answer words). Bidirectional LSTMs are employed to encode the input into contextualized representations. Besides two standard elements, namely attention and copy, the decoder contains a special initialization to control the difficulty of the generated question. Specifically, we map the difficulty label d into a global difficulty variable with a lookup table, and combine the variable with the last hidden state of the encoder to initialize the decoder.

Table 3.1: Distance statistics for non-stop words.

	Easy	Hard	All
Avg. distance of question words	7.67	9.71	8.43
Avg. distance of all sentence words	11.23	11.16	11.20

3.4.1 Exploring Proximity Hints

Recall that our first intuition tells that the proximity hints are helpful for answering the SQuAD-like questions. Before introducing our design for implementing the intuition, we quantitatively verify it by showing some statistics. Specifically, we examine the average distance of those nonstop question words that also appear in the input sentence to the answer fragment. For example, for Q1 in Figure 3.1 and its corresponding input sentence "Oxygen is a chemical element with symbol O and atomic number 8", we calculate the word-level average distance of words "atomic", "number", "element", and "oxygen" to the answer "8". The statistics are given in Table 3.1. In contrast, the average distance of all nonstop sentence words to the answer is also given in the bottom line. If we only count those nonstop question words, we find that their distance to the answer fragment is much smaller than the sentence words, namely 8.43 vs. 11.20. We call this *Question Word Proximity Hint* (**QWPH**). More importantly, the distance for hard questions is significantly larger than that for easy questions, namely 9.71 vs. 7.67, which well verifies our intuition that if a question has more obvious proximity hints (i.e., containing more words that are near the answer in the corresponding sentence), it would be easier to solve. We model QWPH for easy questions and hard questions separately and call this *Difficulty Level Proximity Hint*

(**DLPH**).

To implement the QWPH intuition, our model learns a lookup table which maps the distance of each sentence word to the answer fragment, i.e., 0 (for answer words), 1, 2, etc., into a position embedding: $(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_L)$, where $\mathbf{p}_i \in \mathbb{R}^{d_p}$ and d_p is the dimension. L is the maximum distance we consider. Different from QWPH, the DLPH intuition additionally explores the information of question difficulty levels. Therefore, we define two lookup tables: $(\mathbf{p}^e_0, \mathbf{p}^e_1, \mathbf{p}^e_2, ..., \mathbf{p}^e_L)$ for the Easy label, and $(\mathbf{p}^h_0, \mathbf{p}^h_1, \mathbf{p}^h_2, ..., \mathbf{p}^h_L)$ for the Hard label. Note that the above position embeddings not only carry the information of sentence word position, but also let our model know which aspect (i.e., answer) to ask with the embeddings of position 0.

3.4.2 Characteristic-rich Encoder

The characteristic-rich encoder incorporates several features into a contextualized representation. For each sentence word w, an embedding lookup table is firstly used to map tokens in the sentence into dense vectors: $(\mathbf{w}_1,\ \mathbf{w}_2, ..., \mathbf{w}_m)$, where $\mathbf{w}_i \in \mathbb{R}^{d_w}$ of d_w dimensions. Then we concatenate its word embedding and position embedding (proximity hints) to derive a characteristic-rich embedding: $\mathbf{x} = [\mathbf{w}; \mathbf{p}]$. We use bidirectional LSTMs to encode the sequence $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m)$ to get a contextualized representation for each token:

$$\overrightarrow{\mathbf{h}}_i = \overrightarrow{\mathrm{LSTM}}(\overrightarrow{\mathbf{h}}_{i-1}, \mathbf{x}_i),\ \ \overleftarrow{\mathbf{h}}_i = \overleftarrow{\mathrm{LSTM}}(\overleftarrow{\mathbf{h}}_{i+1}, \mathbf{x}_i),$$

where $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ are the hidden states at the i-th time step of the forward and the backward LSTMs. We concatenate them together as $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]$.

3.4.3 Difficulty-controllable Decoder

We use another LSTM as the decoder to generate the question. We employ the difficulty label d to initialize the hidden state of the decoder. During the decoding, we incorporate the attention and copy mechanisms to enhance the performance.

Global Difficulty Control. We regard the generation of difficulty-controllable questions as a problem of sentence generation towards a specified style, i.e., easy or hard. To do so, we introduce a global difficulty variable to control the generation. We follow the recent works for the task of style transfer that apply the control variable globally, i.e., using the style variable to initialize the decoder [44]. Specifically, for the specified difficulty level d, we first map it to its corresponding difficulty variable $\mathbf{d} \in \mathbb{R}^{d_d}$, where d_d is the dimension of a difficulty variable. Then we use the concatenation of $\mathbf{d}$ with the final hidden state $\mathbf{h}_m$ of the encoder to initialize the decoder hidden state $\mathbf{u}_0 = [\mathbf{h}_m; \mathbf{d}]$. Note that in the training stage, we feed the model the ground truth difficulty labels, while in the testing stage, our model can take any specified difficulty labels, i.e., difficulty-controllable, for question generation. We have also tried some variations by adding this variable to other places such as every encoder or decoder input in the model but it does not work.

Decoder with Attention & Copy. The decoder predicts the word probability distribution at each decoding timestep to generate the question. At the t-th timestep, it reads the word embedding $\mathbf{w}_t$ and the hidden state $\mathbf{u}_{t-1}$ of the previous timestep to generate the current hidden state $\mathbf{u}_t = \text{LSTM}(\mathbf{u}_{t-1}, \mathbf{w}_t)$. Then the decoder employs the attention mechanism [51] and copy mechanism [71] to generate the question by copying words in the sentence or generating words from a predefined vocabulary.

3.4.4 Training and Inference

In the training, our model minimizes the following negative log-likelihood of all training instances:

$$\mathcal{L} = -\sum_{q \in \mathcal{Q}} \log \mathrm{P}(q|a, s, d), \tag{3.1}$$

where $\mathcal{Q}$ includes all training data points, and $\log \mathrm{P}(q|a, s, d)$ is the conditional log-likelihood of q. For testing, we can generate questions of diverse difficulty levels $d_i \in \mathcal{D}$ (predefined difficulty levels) by maximizing:

$$\overline{q} = \arg\max_{q} \log \mathrm{P}(q|a, s, d_i). \tag{3.2}$$

3.5 Experiments

3.5.1 Experimental Settings

Dataset. Our prepared dataset is split according to articles of the SQuAD data, and Table 3.2 provides the detailed statistics. Across the training, validation and test sets, the splitting ratio is around 7:1:1, and the easy sample ratio is around 58% for all three.

Baselines and Ablation Tests. We only employ neural network based methods as our baselines, since they perform better than non-neural methods as shown in recent works [15, 108]. The first

Table 3.2: The statistics of our dataset.

	Train	Dev	Test
# easy questions	34,813	4,973	4,937
# hard questions	24,317	3,573	3,442
Easy ratio	58.88%	58.19%	58.92%

baseline models the question generation as a seq2seq problem incorporating the attention mechanism, and we refer to it as **L2A** [15]. The second baseline **Ans** adds answer indicator embedding to the seq2seq model, similar to [108]. Two ablations that only employ the question word proximity hint or the difficulty level proximity hint are referred to as **QWPH** and **DLPH**. Moreover, we examine the effectiveness of the global difficulty control (**GDC**) combined with QWPH and DLPH, refer to them as **QWPH-GDC** and **DLPH-GDC**. All these methods are enhanced by the *copy* mechanism.

Model Details and Parameter Settings. The embedding dimensions for the position embedding and the global difficulty variable, i.e. d_p and d_d, are set to 50 and 10 respectively. We use the maximum relative distance $L = 20$ in the position embedding. We adopt teacher-forcing in the encoder-decoder training and use the ground truth difficulty labels. In the testing procedure, we select the model with the lowest perplexity and beam search with size 3 is employed for question generation. All important hyper-parameters, such as d_p and d_d, are selected on the validation dataset.

Table 3.3: Difficulty of the generated questions, measured with R-Net and BiDAF. For easy questions, higher score indicates better difficulty-control, while for hard questions, lower indicates better.

	Easy Questions Set				**Hard** Questions Set			
	R-Net		BiDAF		R-Net		BiDAF	
	EM	F1	EM	F1	EM	F1	EM	F1
Ans	82.16	87.22	75.43	83.17	34.15	60.07	29.36	55.89
QWPH	82.66	87.37	76.10	83.90	33.35	59.50	28.40	55.21
QWPH-GDC	84.35	88.86	77.23	84.78	31.60	57.88	26.68	54.31
DLPH	85.49	89.50	78.35	85.34	28.05	54.21	24.89	51.25
DLPH-GDC	**85.82**	**89.69**	**79.09**	**85.72**	**26.71**	**53.40**	**24.47**	**51.20**

3.5.2 Difficulty Control Results

We run R-Net and BiDAF to assess the difficulty of our generated hard and easy questions. Here the R-Net and BiDAF systems are trained using the same train/validation splits as shown in Table 3.2, and we report their performance under the standard reading comprehension measures for SQuAD questions, i.e., Exact Match (**EM**) and macro-averaged F1 score (**F1**), on the easy and hard question sets respectively. For all experiments, we firstly show the performance of difficulty-controllable question generation by feeding ground truth difficulty labels, then we feed the reverse difficulty labels to demonstrate our model can ***control*** the difficulty of generated questions.

Recall that the generated questions can be split into an easy set and a hard set according to the difficulty labels. Here we evaluate the generated questions from the perspective that a reading comprehension system (e.g., R-Net and BiDAF) should perform better on the generated questions in the easy set, and perform worse on the hard question set. If a pipeline does not use the answer information, its generated questions are likely not about the answers, thus both BiDAF and R-Net cannot work well no matter for easy or hard questions. Therefore, we do not use L2A here.

As shown in Table 3.3, for the easy set, the questions generated by the methods using the difficulty label "Easy" are easier to answer. Specifically, compared with Ans and QWPH which cannot control the difficulty, QWPH-GDC, DLPH, and DLPH-GDC generate easier questions, showing that they have the capability of generating difficulty-controllable questions. One instant doubt is that a model can simply produce trivial questions by having them contain the answer words. In fact, our models do not have this behaviour, because it will increase the training loss. To further verify this, we calculate the occurrence rate of answer words in the generated questions. The result shows that only 0.09% answer words appear in the questions generated by our models.

For the hard set, we can draw the same conclusion by keeping in mind that a lower score indicates the corresponding method performs better in generating difficulty-controllable questions. (Note that questions irrelevant to the answer can also yield lower scores, and we have more discussion about this issue in Section 3.5.3 for the human evaluation.) This observation shows that incorporating the difficulty information locally by the two position embeddings or globally by the difficulty-controlled initialization indeed guides the generator to generate easier or harder questions. Comparing DLPH and QWPH-GDC, we find that the local difficulty control by the position embedding is more effective. DLPH-GDC performs the best by combining the local and global difficulty control signals.

Table 3.4: The results of controlling difficulty, measured with R-Net and BiDAF. The scores are performance gap between questions generated with original difficulty label and questions generated with reverse difficulty label.

	Easy Questions Set				**Hard** Questions Set			
	R-Net		BiDAF		R-Net		BiDAF	
	`EM`	`F1`	`EM`	`F1`	`EM`	`F1`	`EM`	`F1`
QWPH-GDC	7.41	5.72	7.13	5.88	6.45	5.47	6.13	5.10
DLPH	12.41	9.51	11.28	8.49	12.01	10.45	10.51	9.37
DLPH-GDC	**12.91**	**9.95**	**12.40**	**9.23**	**12.68**	**10.76**	**11.22**	**9.97**

Moreover, we find that QWPH achieves slightly better performance than Ans baseline. A large performance gap between QWPH-GDC and QWPH again validates the effectiveness of the global difficulty control. Meanwhile, the improvement from QWPH to DLPH shows that the local difficulty level proximity hint can stress the question difficulty at each time step to perform better.

On the other hand, another way to validate our model is testing whether our model can ***control*** the difficulty by feeding the reversed difficulty labels. For example, for a question in the easy set, if we feed the "Hard" label together with the input sentence and answer of this question into our model, we expect the generated question should be harder than feeding the "Easy" label. Concretely, if a method has the better capability in controlling the difficulty, on two sets of questions generated with this method by taking the true label and the reversed label, the performance gap of a reading comprehension system should be larger. The results of this experiment are given in Table 3.4. We only compare models which have difficulty control capability. The model combining local and global difficulty signals, i.e., DLPH-GDC, achieves the largest

Table 3.5: Human evaluation results for generated questions. `Fluency(F)` and `Difficulty(D)` take values from {1, 2, 3} (3 means the top fluency or difficulty), while `Relevance(R)` takes a binary value, i.e., 1 or 0.

	Easy Question Set			**Hard** Question Set		
	`F`	`D`	`R`	`F`	`D`	`R`
Ans	2.91	2.02	0.74	2.87	2.12	0.58
DLPH-GDC	2.94	1.84	0.76	2.87	2.26	0.64

gap, which again shows that: (1) DLPH-GDC has the strongest capability of generating difficulty-controllable questions; (2) The local difficulty control (i.e. DLPH) is more effective than the global (i.e. QWPH-GDC).

3.5.3 Manual Evaluation

We hire 3 annotators to rate the model generated questions. We randomly sampled 100 questions with "Easy" labels and 100 with "Hard" labels from the test set, and let each annotator annotate these 200 cases. During the annotation, each data point contains a sentence, an answer, and the questions generated by different models, without showing the difficulty labels. We consider three metrics: `Fluency(F)`, `Difficulty(D)` and `Relevance(R)`. The annotators are first asked to read the generated questions to evaluate their grammatical correctness and fluency. Then, all annotators are required to rate the difficulty of each generated question by considering the corresponding sentence and answer. Finally, for relevance, we ask the annotators to judge if the question is asking about the answer. `Fluency` and `Difficulty` take values from {1, 2, 3} (3 means the top fluency or difficulty), while `Relevance` takes a binary value (1 or 0).

Table 3.5 shows the results of the manual evaluation. We compare our best model DLPH-GDC with the Ans baseline. We separate the `Easy` questions and `Hard` questions for statistics. For both question sets, both models achieve high scores on `Fluency`, owing to the strong language modeling capability of neural models. For `Difficulty`, we can find that DLPH-GDC can generate easier or harder questions than Ans by feeding the true difficulty labels. Another observation is that, for the Ans baseline, questions generated in the `Easy` set are easier than those in the `Hard` set, which validates our difficulty labelling protocol from another perspective. Note that for human beings, all SQuAD-like questions are not really difficult, therefore, the difference of `Difficulty` values between the easy set and the hard set is not large.

Furthermore, we can observe our model can generate more relevant questions compared with the Ans baseline. The reason could be that our position embedding can not only tell where the answer words are, but also indicate the distance of the context words to the answer. Thus, it provides more information to the model for asking to the point questions. Ans only differentiates the answer token and non-answer token, and treats all non-answer tokens equally.

Recall that we had the concern regarding Table 3.3 that the generated hard questions by our difficulty-controlling models say DLPH-GDC may simply be irrelevant to the answer, which makes DLPH-GDC achieves lower EM/F1 scores than the Ans baseline. By comparing the `Relevance` scores in Table 3.5 and EM/F1 scores in Table 3.3 for Hard Question Set, we find that the questions generated by DLPH-GDC are more relevant

Table 3.6: Automatic evaluation for question quality.

	B1	B2	B3	B4	MET	R-L
L2A	36.01	21.61	14.97	10.88	15.99	38.06
Ans	43.51	29.06	21.35	16.22	20.53	45.66
QWPH	43.75	29.28	21.61	16.46	20.70	46.02
QWPH-GDC	43.99	29.60	21.86	16.63	20.87	46.26
DLPH	44.11	29.64	21.89	16.68	20.94	46.22
DLPH-GDC	43.85	29.48	21.77	16.56	20.79	46.16

(as shown in Table 3.5) and more difficult (as shown in both Tables 3.3 and 3.5) than those generated by the Ans baseline. This observation resolves our doubt on the irrelevance issue and supports the conclusion that our DLPH-GDC does generate more difficult and relevant questions which can fail the two RC pipelines.

3.5.4 Automatic Evaluation of Question Quality

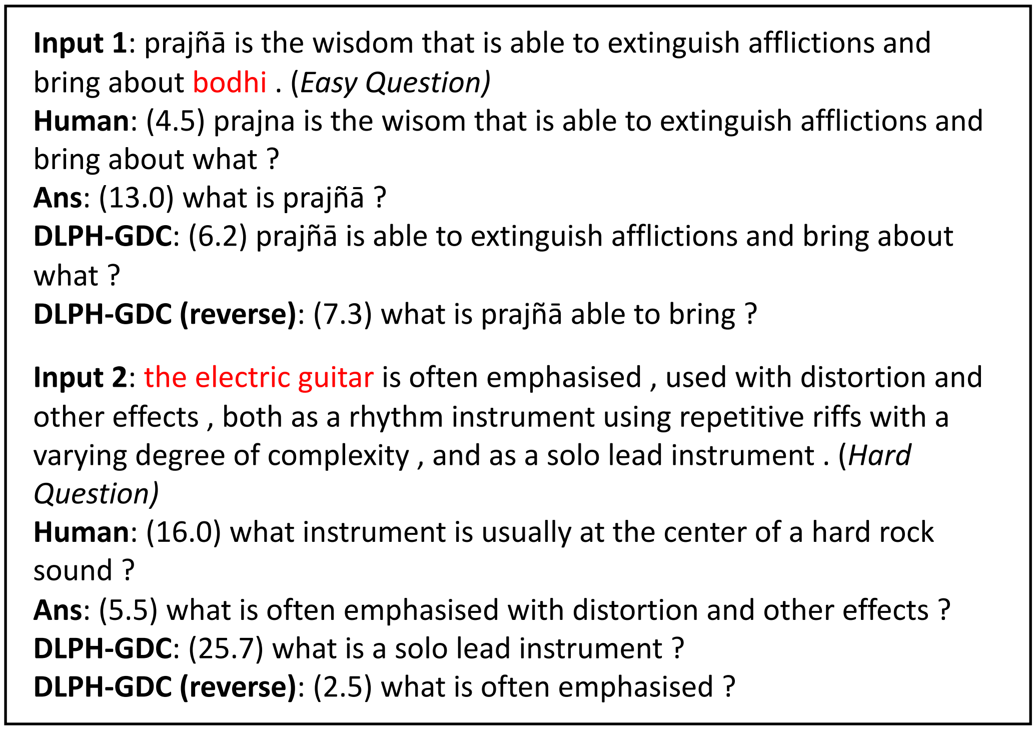
Here we evaluate the similarity of generated questions with the ground truth. Since our dataset is not parallel (i.e., for a sentence and answer pair, our dataset only has one question with the "easy" or "hard" label), here we only evaluate the question quality by feeding the ground truth difficulty labels. We employ BLEU (B), METEOR (MET) and ROUGE-L (R-L) scores by following [15]. BLEU evaluates the average N-gram precision on a set of reference sentences, with a penalty for overly long sentences. ROUGE-L is commonly employed to evaluate the recall of the longest common subsequences, with a penalty for short sentences.

Table 3.6 shows the quality of generated questions. Comparing the first three methods, we can find that the answer and position information helps a lot for asking to the point questions, i.e., more similar to the ground truth. Moreover, QWPH performs better than Ans, indicating that further distinguishing the different distance of the non-answer words to the answer provides richer information for the model to generate better questions. The results in the lower half show that, given the ground truth difficulty labels, these three methods with the capability of difficulty control are better than the first three methods. These three models achieve comparable performance, and DLPH-GDC sacrifices a little in N-gram based performance here while achieving the best difficulty control capability (refer to Tables 3.3 & 3.4).

3.5.5 Case Study

Figure 3.3 provides some examples of generated questions (with answers marked in red). The number after the model is the average distance of the overlapped nonstop words between the question and the input sentence to the answer fragment. The average distance corresponds to the our intuition proximity hints well. Compared with questions generated by Ans baseline, our model can give more hints (shorter distance) when asking easier questions and give less hints (longer distance) when asking harder questions.

For the first example, we observe that the ground truth question generated by Human is quite easy, just replacing the answer “bodhi” with “what”. Among the three systems, Ans asks a question that is not about the answer. While both DLPH-GDC and DLPH-GDC (reverse) are able to generate to the point



Input 1: prajñā is the wisdom that is able to extinguish afflictions and bring about bodhi . (*Easy Question)*
Human: (4.5) prajna is the wisom that is able to extinguish afflictions and bring about what ?
Ans: (13.0) what is prajñā ?
DLPH-GDC: (6.2) prajñā is able to extinguish afflictions and bring about what ?
DLPH-GDC (reverse): (7.3) what is prajñā able to bring ?

Input 2: the electric guitar is often emphasised , used with distortion and other effects , both as a rhythm instrument using repetitive riffs with a varying degree of complexity , and as a solo lead instrument . (*Hard Question)*
Human: (16.0) what instrument is usually at the center of a hard rock sound ?
Ans: (5.5) what is often emphasised with distortion and other effects ?
DLPH-GDC: (25.7) what is a solo lead instrument ?
DLPH-GDC (reverse): (2.5) what is often emphasised ?

Figure 3.3: Example questions (with answers marked in red). The human question for Input 2 uses some information ("hard rock") in preceding sentences which are not shown here.

questions. Specifically, by taking the “Easy” label, DLPH-GDC tends to use more words from the input sentence, while DLPH-GDC (reverse) uses less and its generated question is relatively difficult. For the second example, we find our system is also applicable to the question with “Hard” label.

3.6 Summary

In this chapter, we present a novel setting, namely difficulty-controllable question generation for reading comprehension, which to the best of our knowledge has never been studied before. We propose an end-to-end approach to learn the question generation with designated difficulty levels. We also prepared the first dataset for this task, and extensive experiments show that our framework can solve this task reasonably well. One interesting future direction is to explore generating multiple questions for different aspects in one sentence.

□ **End of chapter.**

Chapter 4

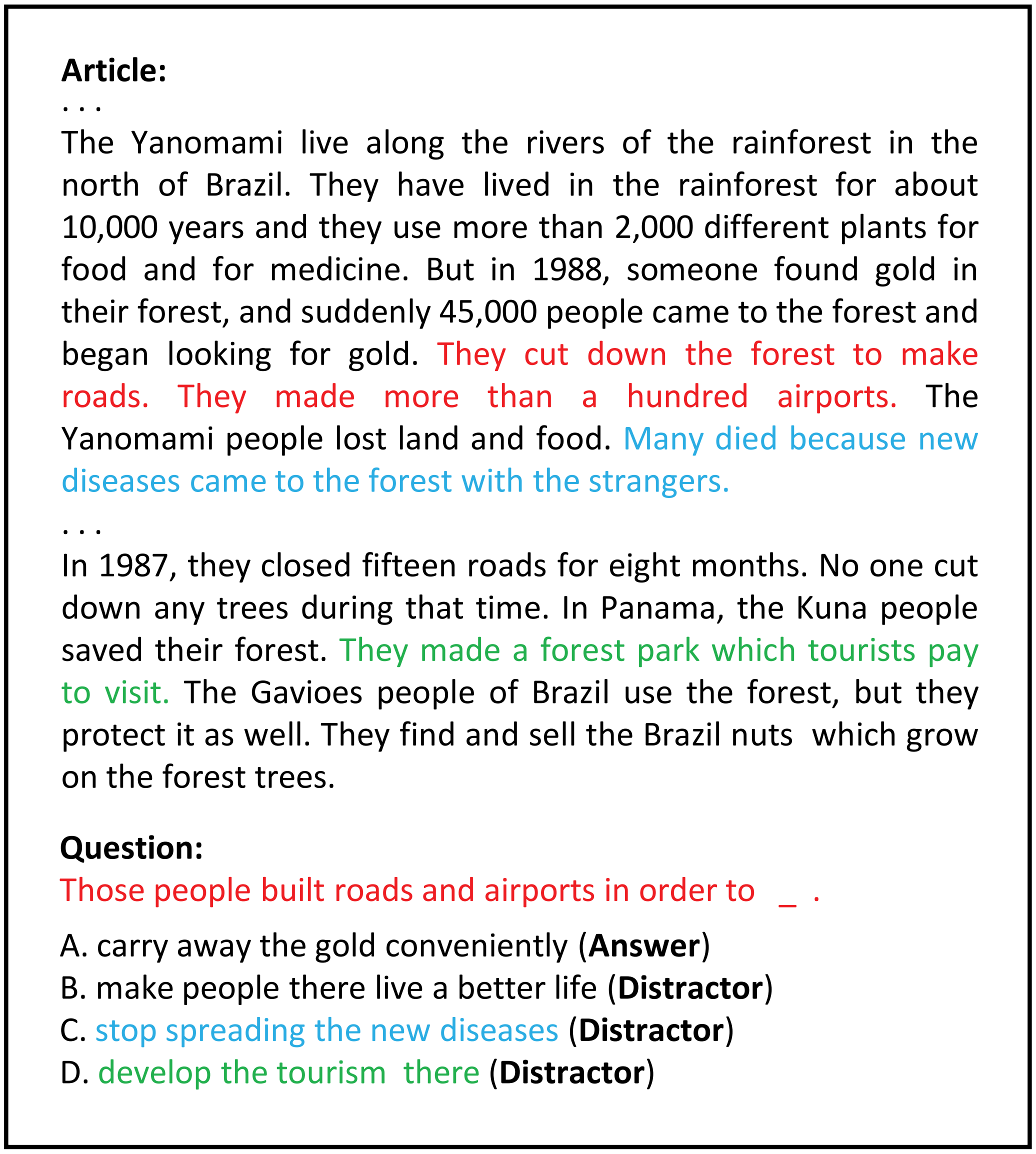
Distractor Generation for Multiple Choice Questions

In this chapter, we investigate the task of distractor generation for multiple-choice reading comprehension questions from examinations. In contrast to all previous works, we do not aim at preparing words or short phrases distractors. Instead, we endeavor to generate longer and semantic-rich distractors which are closer to distractors in real reading comprehension from examinations. We first give an introduction and describe the preliminary in Section 4.1. In Section 4.2, we define the distractor generation task and propose a hierarchical encoder-decoder framework with static and dynamic attention mechanisms to tackle this task. Section 4.3 and Section 4.4 describe our experimental setting and results. Finally, we give a summary of this chapter in Section 4.5.

4.1 Introduction

Reading comprehension (RC) is regarded as an avant-garde task in NLP research for practising the capability of language understanding. Models with recent advances of deep learning techniques are even capable of exceeding human performance in some RC tasks, such as for questions with span-based answers [103]. However, it is not the case when directly applying the state-of-the-art models to multiple choice questions (MCQs) in RACE dataset [35], elaborately designed by human experts for real examinations, where the task is to select the correct answer from a few given options after reading the article. The performance gap between the state-of-the-art deep models (53.3%) [86] and ceiling (95%) [35] is significant. One possible reason is that in MCQs, besides the *question* and the correct *answer* option, there are a few *distractors* (wrong options) to distract humans or machines from the correct answer. Most distractors are somehow related to the answer and consistent with the semantic context of the question, and all of them have correct grammar [21, 42]. Furthermore, most of the distractors have some trace in the article, which fails the state-of-the-art models utilizing context matching only to yield decent results.

The MCQs in the RACE dataset are collected from the English exams for Chinese students from grade 7 to 12. Constructing RACE-like MCQ dataset is important and nontrivial, because poor distractor options can make the questions almost trivial to solve [94] and reasonable distractors are time-consuming to design. In this chapter, we investigate the task of automatic *distractor generation* (DG) . The task aims to generate reasonable distractors for RACE-like MCQs, given a reading comprehension article, and a pair of question and its correct answer originated from the article. Figure 4.1 shows an example multiple choice question with four options. We can find that all options are grammatically coherent with the



Article:

. . .

The Yanomami live along the rivers of the rainforest in the north of Brazil. They have lived in the rainforest for about 10,000 years and they use more than 2,000 different plants for food and for medicine. But in 1988, someone found gold in their forest, and suddenly 45,000 people came to the forest and began looking for gold. They cut down the forest to make roads. They made more than a hundred airports. The Yanomami people lost land and food. Many died because new diseases came to the forest with the strangers.

. . .

In 1987, they closed fifteen roads for eight months. No one cut down any trees during that time. In Panama, the Kuna people saved their forest. They made a forest park which tourists pay to visit. The Gavioes people of Brazil use the forest, but they protect it as well. They find and sell the Brazil nuts which grow on the forest trees.

Question:

Those people built roads and airports in order to _ .

A. carry away the gold conveniently (**Answer**)
B. make people there live a better life (**Distractor**)
C. stop spreading the new diseases (**Distractor**)
D. develop the tourism there (**Distractor**)

Figure 4.1: Sample multiple choice question along with the corresponding article. The question, options and their relevant sentences in the article are marked with the same color.

question, and semantically relevant to the article. Distractor generation is of great significance in a few aspects. It can aid the preparation of MCQ reading comprehension datasets. With large datasets prepared, it is expectable that the performance of reading comprehension systems for MCQs will be boosted, as we have observed such improvements [100] by applying generated question-answer pairs to train models to solve SQuAD questions. It could also be helpful to alleviate instructors' workload in designing MCQs for students.

Automatic DG is different from previous distractor preparation works, which basically follow an extraction-selection manner. First, a distractor candidate set is extracted from multiple sources, such as GloVe vocabulary [61], noun phrases from textbooks [94] and articles [1]. Then similarity based [22, 77, 33, 56] or learning based [42, 70, 43] algorithms are employed to select the distractors. Another manner is to apply some pre-defined rules to prepare distractors by changing the surface form of some words or phrases [5]. Automatic DG for RACE-like MCQs is a challenging task. First, different from previous works that prepare word or short phrase distractors (1.46 tokens on average in SciQ [94]), we here endeavor to generate longer and semantic-rich distractors. Specifically, the average length of the distractors in our experimental dataset is 8.1. Furthermore, the generated distractors should semantically related to the reading comprehension question, since it is trivial to identify a distractor having no connection with the article or question. Moreover, the distractors should not be paraphrases of the correct answer option. Finally, the generated distractors should be grammatically consistent with the question, especially for questions with a blank in the end, as shown in Figure 4.1.

Previous works following the extraction-selection manner cannot meet these requirements.

We formulate the task of automatic distractor generation as a sequence-to-sequence learning problem that directly generates the distractors given the article, and a pair of question and its correct answer. We design our framework to explicitly tackle the above mentioned challenges by using a data-driven approach to learn to meet these requirements automatically. More specifically, we employ the hierarchical encoder-decoder network, which has already shown potentials to tackle long sequential input [84, 48], as the base model for building our framework. On top of the hierarchical encoding structure, we propose the dynamic attention mechanism to combine sentence-level and word-level attentions varying at each recurrent time step to generate a more readable sequence. Furthermore, a static attention mechanism is designed to modulate the dynamic attention not to focus on question-irrelevant sentences or sentences which contribute to the correct answer option. Finally, we use a question-based initializer as the start point to generate the distractor, which makes the distractor grammatically consistent with the question. In the generation stage, we use the beam search to generate three diverse distractors by controlling their distance.

In the evaluations, we conduct experiments on a distractor generation dataset prepared from RACE using n-gram based automatic evaluation metrics such as BLEU and ROUGE. The results show that our proposed model beats several baselines and ablations. Human evaluations show that distractors generated by our model are more likely to confuse the examinees, which demonstrates the functionality of our generated distractors in real examinations.

4.2 Framework Description

4.2.1 Task Definition

In the task of automatic Distractor Generation (DG), given an article, a pair of question and its correct option originated from the article, our goal is to generate context and question related, grammatically consistent wrong options, i.e. distractor, for the question.

Formally, let P denote the input article containing multiple sentences: $s_1, s_2, ..., s_n$, q and a denote the question and its correct answer, respectively. The DG task is defined as finding the distractor $\overline{d}$, such that:

$$\overline{d} = \arg\max_{d} \log \mathrm{P}(d|P, a, q), \tag{4.1}$$

where $\log \mathrm{P}(d|P, a, q)$ is the conditional log-likelihood of the predicted distractor d, give P, a and q.

4.2.2 Framework Overview

A straightforward strategy for distractor generation is to employ the standard sequence-to-sequence learning network [82] to learn the mapping from the article to the distractor. Unfortunately, an article can be too long as the input, which cannot receive decent results. Here we advocate the hierarchical encoder-decoder framework to model such long sequential input. The architecture of our overall framework is depicted in Figure 4.2.

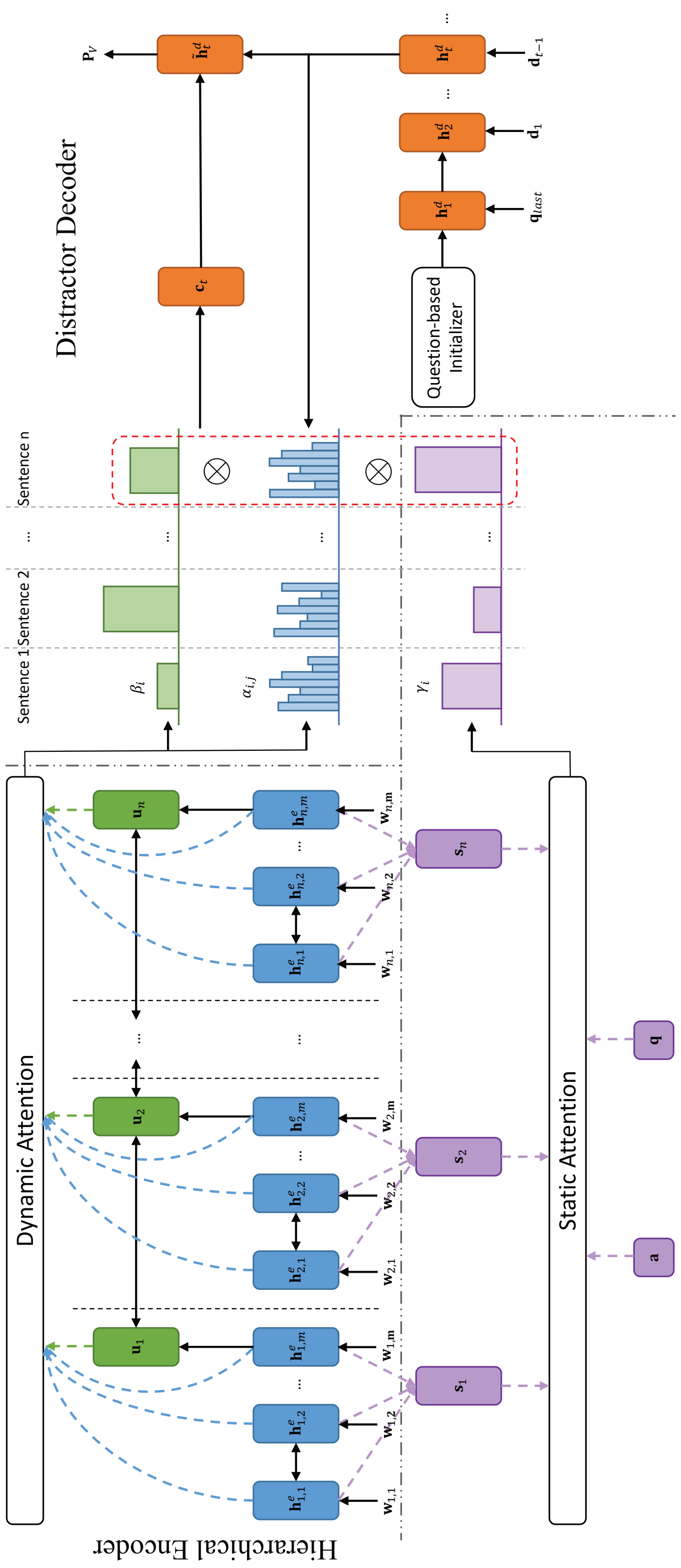


Figure 4.2: A overview of our model that jointly utilizes static and dynamic attentions. *(Better viewed in color)*.

First, we employ the hierarchical encoder to obtain hierarchical contextualized representations for the whole article, namely, word-level representation and sentence-level representation. Before decoding the encoded information, we design a static attention mechanism to model the global sentence importance considering the fact that the distractor should be semantically related to the question and should not share the same semantic meaning with the correct answer. The static attention distribution is used in the decoder as a soft gate to modulate the dynamic attention. For the decoder part, we first employ a language model to compress the question information into a fixed length vector to initialize the decoder state, making the distractor grammatically consistent with the question. During each decoding step, the dynamic hierarchical attention combines the sentence-level and word-level information to attend different part at each decoding time step. With the combined architecture, our model can generate grammatically consistent, context and question related wrong options (distractors) in an end-to-end manner.

4.2.3 Hierarchical Encoder

Word Embedding. An embedding lookup table is firstly used to map tokens in each sentence s_i in the article P into word dense vectors $(\mathbf{w}_{i,1}, \mathbf{w}_{i,2}, ..., \mathbf{w}_{i,m})$, where $\mathbf{w}_{i,j} \in \mathbb{R}^{d_w}$ having d_w dimensions.

Word Encoder. For each sentence s_i, the word encoder takes its word vectors $(\mathbf{w}_{i,1}, \mathbf{w}_{i,2}, ..., \mathbf{w}_{i,m})$ as input. Specifically, we use bidirectional LSTMs to encode the sequence to get a contextualized representation for each word:

$$\overrightarrow{\mathbf{h}^e_{i,j}} = \overrightarrow{\text{LSTM}}(\overrightarrow{\mathbf{h}^e_{i,j-1}}, \mathbf{w}_{i,j}),\ \overleftarrow{\mathbf{h}^e_{i,j}} = \overleftarrow{\text{LSTM}}(\overleftarrow{\mathbf{h}^e_{i,j+1}}, \mathbf{w}_{i,j}),$$

where $\overrightarrow{\mathbf{h}^e_{i,j}}$ and $\overleftarrow{\mathbf{h}^e_{i,j}}$ are the hidden states at the j-th time step of the forward and the backward LSTMs. We concatenate them together as $\mathbf{h}^e_{i,j} = [\overrightarrow{\mathbf{h}^e_{i,j}}; \overleftarrow{\mathbf{h}^e_{i,j}}]$.

Sentence Encoder. On top of the word encoding layer, we combine the final hidden state of the forward LSTM and the first hidden state of the backward LSTM of each sentence as the sentence representation and employ another bidirectional LSTMs to learn the contextual connection of sentences. We denote the contextualized representation of the sentence sequence as $(\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n)$.

4.2.4 Static Attention Mechanism

Recall that the generated distractors should be semantically relevant to the question, but must not share the same semantic meaning with the answer. To achieve this goal, here we introduce a static attention mechanism which learns an importance distribution $(\gamma_1, \gamma_2, ..., \gamma_n)$ of the sentences $(s_1, s_2, ..., s_n)$ in the article. Here we use the answer a and the question q as queries to interact with all sentences to learn such distribution.

Encoding Layer. In the encoding layer, we transform the answer a, the question q and all sentences $(s_1, s_2, ..., s_n)$ into fixed length vector representations. Specifically, two individual bidirectional LSTM networks are employed to encode a and q separately to derive the contextualized representation for each token in them and obtain $(\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_k)$ and $(\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_l)$, respectively. Then an average pooling layer is employed to acquire the representation for the question and answer:

$$\mathbf{a} = \frac{1}{k}\sum_{t=1}^{k}\mathbf{a}_t, \mathbf{q} = \frac{1}{l}\sum_{t=1}^{l}\mathbf{q}_t. \tag{4.2}$$

For the sentence representation, we do not reuse the sentence representation $\mathbf{u}_i$ from the sentence encoder since $\mathbf{u}_i$ is responsible for learning the semantic information for a whole sentence, while here we only want to learn the importance distribution of sentences according to the query (i.e. a pair of question and answer). Therefore, we only reuse the word-level contextualized representations $\mathbf{h}^e_{i,j}$ learned in the hierarchical encoder and employ the same average pooling layer to get the representation of each sentence:

$$\mathbf{s}_i = \frac{1}{m}\sum_{t=1}^{m}\mathbf{h}^e_{i,t}. \tag{4.3}$$

Matching Layer. For generating non-trivial distractors, we should emphasize the sentences that are relevant to the question, and suppress the sentences relevant to the answer. For this reason, we learn a score o_i for s_i that combines the above two aspects with bilinear transformation:

$$o_i = \lambda_q \mathbf{s}_i^\top \mathbf{W}_m \mathbf{q} - \lambda_a \mathbf{s}_i^\top \mathbf{W}_m \mathbf{a} + \mathbf{b}_m, \tag{4.4}$$

where $\mathbf{W}_m$ and $\mathbf{b}_m$ are learnable parameters.

Normalization Layer. Before feeding the raw sentence importance score o_i into the Softmax function to compute the final static attention distribution, we use the question to learn a temperature $\tau \in (0, 1)$:

$$\tau = \text{sigmoid}(\mathbf{w_q}^\top \mathbf{q} + b_q), \tag{4.5}$$

where $\mathbf{w_q}$ and b_q are learnable parameters. Then, we derive the static attention distribution as:

$$\gamma_i = \text{softmax}(o_i/\tau). \tag{4.6}$$

The intuition behind using the temperature τ is that if a question asks for some specific details in the article, it is only relevant to one or two sentences. While if a question requires summarizing or reasoning, it could be relevant to many sentences in the article. Therefore, we propose the above data-driven approach to learn the temperature τ according to the property of the question. If τ is close to 0, then it works together with o_i to yield a peaked distribution γ which simulates the case of detailed questions. Otherwise, if τ is close to 1, it will not peak any sentence attention score γ_i.

4.2.5 Distractor Decoder

We use another LSTMs as the decoder to generate the distractor. Instead of using the last hidden state of the encoder to initialize the decoder, we design a special question-based initializer to make the distractor grammatically consistent with the question. During the decoding, we introduce the dynamic attention mechanisms to combine the sentence-level and word-level attentions varying at each recurrent time step to generate a more readable sequence. We also incorporate the static attention here to modulate the dynamic attention to ensure the semantic relevance of the generated distractors.

Question-based Initializer. We design a question-based initializer to initialize the initial state of the decoder. Specifically, we use a question LSTM to encode the question, and use the last time step information of the LSTM in the following manner:

- Instead of using BOS (i.e. the *Begin of Sentence* indicator), we use the last token in the question ($\mathbf{q}_{last}$) as the initial input of the decoder.
- Other than using the final state of the hierarchical encoder to initialize the decoder, we here use the final cell state and hidden state of the question LSTM to initialize the decoder.

Dynamic Hierarchical Attention Mechanism. The standard attention mechanism treats an article as a long sequence and compares the hidden state of the current decoding time step to all encoder hidden states. This approach is not suitable for long input sequences for the following reasons. First, the standard LSTM cannot model such long inputs (on average, 343.9 words per article in our training set). Second, we will lose the sentence structure if we treat the tokens of different sentences equally. Last but not least, usually a question or a distractor is only related to a small number of sentences in the article, we should only use the related sentences to generate the distractor, but the standard attention has no emphasis on difference sentences.

Given the above reasons, we employ the dynamic hierarchical attention to only focus on important sentences during each decoding time step. We call it *dynamic* because both word-level and sentence-level attention distributions change at each time step. When generating a word at the time step t, the decoder reads the word embedding $\mathbf{d}_{t-1}$ and the hidden state $\mathbf{h}^d_{t-1}$ of the previous time step to generate the current hidden state $\mathbf{h}_t^d = \mathrm{LSTM}(\mathbf{h}_{t-1}^d, \mathbf{d}_{t-1})$. Then it calculates both the sentence-level attention β_i and the word-level attention $\alpha_{i,j}$ at the same time:

$$\beta_i = \mathbf{u}_i^\top \mathbf{W}_{d_1} \mathbf{h}_t^d, \quad \alpha_{i,j} = {\mathbf{h}_{i,j}^e}^\top \mathbf{W}_{d_2} \mathbf{h}_t^d, \tag{4.7}$$

where $\mathbf{W}_{d_1}$ and $\mathbf{W}_{d_2}$ are trainable parameters. The sentence-level attention determines how much each sentence should contribute to the generation at the current time step, while the word-level attention determines how to distribute the attention over words in each sentence.

Finally, we use the static attention γ_i to modulate the dynamic hierarchical attention β_i and $\alpha_{i,j}$ by simple scalar multiplication and renormalization. Thus, the combined attention for each token in the article is:

$$\widetilde{\alpha}_{i,j} = \frac{\alpha_{i,j}\beta_i\gamma_i}{\sum_{i,j}\alpha_{i,j}\beta_i\gamma_i}. \tag{4.8}$$

Then the context vector $\mathbf{c}_t$ is derived as a combination of all article token representations reweighted by the final combined attention $\widetilde{\alpha}_{i,j}$:

$$\mathbf{c}_t = \sum_{i,j} \widetilde{\alpha}_{i,j} \mathbf{h}_{i,j}^e. \tag{4.9}$$

And the attentional vector is calculated as:

$$\tilde{\mathbf{h}}_t^d = \tanh(\mathbf{W}_{\tilde{\mathbf{h}}}[\mathbf{h}_t^d; \mathbf{c}_t]). \tag{4.10}$$

Then, the predicted probability distribution over the vocabulary V at the current step is computed as:

$$\mathrm{P}_V = \mathrm{softmax}(\mathbf{W}_V \tilde{\mathbf{h}}_t^d + \mathbf{b}_V), \tag{4.11}$$

where $\mathbf{W}_{\tilde{\mathbf{h}}}$, $\mathbf{W}_V$ and $\mathbf{b}_V$ are learnable parameters.

4.2.6 Training and Inference

Given the training corpus $\mathcal{Q}$ in which each data sample contains a distractor d, an article P, a question q and an answer a, we minimize the negative log-likelihood with respect to all learnable parameters Θ for training:

$$\mathcal{L} = -\sum_{d \in \mathcal{Q}} \log \mathrm{P}(d|P, a, q; \Theta). \tag{4.12}$$

During generation, if UNK (i.e. unknown words) is decoded at any time step, we replace it with the word having the largest attention weight in the article.

Since there are several diverse distractors (2.4 on average according to Table 4.1) corresponding to the same question in our dataset, we use beam search with beam size k in the testing stage and receive k candidate distractors with decreasing likelihood. The ultimate goal is to generate several diverse distractors, however, usually the successive output sequences from beam search would be similar. Therefore we design the following protocol to generate three diverse distractors. Firstly, we select the distractor with the maximum likelihood as d_1^g. Then we select d_2^g among the remaining candidate distractors along the decreasing order of the likelihood, restricting that the Jaccard distance between d_1^g and d_2^g is larger than 0.5. Finally, d_3^g is selected in a similar way where its distances to both of d_1^g and d_2^g are restricted.

4.3 Experimental Settings

Table 4.1: The statistics of our dataset.

# Train Samples	96501
# Dev Samples	12089
# Test Samples	12284
Avg. article length (tokens)	347.0
Avg. distractor length	8.5
Avg. question length	9.9
Avg. answer length	8.7
Avg. # distractors per question	2.1

4.3.1 Dataset

We evaluate our framework on a distractor generation dataset prepared with the RACE [35] dataset. RACE contains 27,933 articles with 97,687 questions from English examinations of Chinese students from grade 7 to 12. We first extract each data sample as a quadruple of article, question, answer and distractor from RACE, followed by some simple preprocessing steps, such as tokenization, sentence splitting, and lower-casing.

After some investigation on the RACE dataset, we observe that some distractors have no semantic relevance with the article, which can be easily excluded in the examination and also do not make sense for the task of distractor generation since our goal is to generate confusing distractors. Hence, we first filter out such irrelevant distractors by simply counting meaningful tokens in individual distractors. We define a token meaningful if it is not a stop word and has a POS tag from {'JJ', 'JJR', 'JJS', 'NN', 'NNP', 'NNPS', 'NNS', 'RB', 'RBR', 'RBS', 'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ'}. Then, we prune the dataset based on the following constraint: For those meaningful tokens in a distractor that also appear in the article, if their total weighted frequency is no less than 5, the distractor will be kept. Here the weighted frequency of a meaningful token means the multiplication of its frequency in the distractor and its frequency in the article. Moreover, we remove the questions which need to fill in the options at the beginning or in the middle of the questions. Table 4.1 reports the statistics of the processed dataset. We randomly divide the dataset into the training (80%), validation (10%) and testing sets (10%).

4.3.2 Implementation Details

We keep the most frequent 50k tokens in the entire training corpus as the vocabulary, and use the `GloVe.840B.300d` word embeddings [61] for initialization and finetune them in the training. Both source and target sides of our model share the same word embedding. All other tokens outside the vocabulary or cannot found in GloVe are replaced by the UNK symbol. We set the number of layers of LSTMs to 1 for the hierarchical encoder (for both word encoder and sentence encoder) and the static attention encoder, and 2 for the decoder. The bidirectional LSTMs hidden unit size is set to 500 (250 for each direction). For the LSTM used in the question-based initialier, we use 2 layers unidirectional LSTMs with hidden size 500. The hyperparameters λ_q and λ_a in static attention are initialized as 1.0 and 1.5 respectively. We use dropout with probability $p = 0.3$. All trainable parameters, except word embeddings, are randomly initialized with $\mathcal{U}(-0.1, 0.1)$. For optimization in the training, we use stochastic gradient descent (SGD) as the optimizer with a minibatch size of 32 and the initial learning rate 1.0 for all baselines and our model. We train the model for 100k steps and start halving the learning rate at step 50k, then we halve the learning rate every 10k steps till ending. We set the gradient norm upper bound to 5 during the training. We employ the teacher-forcing training, and in the generating stage, we set the maximum length for output sequence as 15 and block unigram repeated token, the beam size k is set to 50. All hyperparameters and models are selected on the validation set based on the lowest perplexity and the results are reported on the test set.

4.3.3 Baselines and Ablations

We compare our framework with the following baselines and ablations. **Seq2Seq**: the basic encoder-decoder learning framework [82] with attention mechanism [51]. Here we adopt the global attention with general score function. The hidden size of LSTMs for both encoder and decoder is 500. We select the model with the lowest perplexity on the validation set. **HRED**: the **H**ie**R**archical **E**ncoder-**D**ecoder (HRED) with hierarchical attention mechanism. This architecture has been proven effective in several NLP tasks including summarization [48], headline generation [84], and text generation [40]. Here we keep the LSTMs size as 500 for fairness and set the number of the word encoder and sentence encoder layers as 1 and the decoder layer as 2. We employ the question-based initializer for all baselines to generate grammatically coherent distractors. In the generation stage, we follow the same policy and beam size for baselines and ablations during the inference stage to generate three distractors.

Table 4.2: Automatic evaluation results on all systems by BLEU and ROUGE. 1st, 2nd and 3rd distractors are generated under the same policy. The best performing system for each compound row is highlighted in boldface.

		BLEU$_1$	BLEU$_2$	BLEU$_3$	BLEU$_4$	ROUGE$_1$	ROUGE$_2$	ROUGE$_L$
1st Distractor	Seq2Seq	25.28	12.43	7.12	4.51	14.12	3.35	13.58
	HRED	26.10	13.96	8.83	6.21	14.83	4.07	14.30
	Our Model	**27.32**	**14.69**	**9.29**	**6.47**	**15.69**	**4.42**	**15.12**
2nd Distractor	Seq2Seq	25.13	12.02	6.56	3.93	13.72	3.09	13.20
	HRED	25.18	12.21	6.94	4.40	13.94	3.11	13.40
	Our Model	**26.56**	**13.14**	**7.58**	**4.85**	**14.72**	**3.52**	**14.15**
3rd Distractor	Seq2Seq	25.34	11.53	5.94	3.33	13.78	2.82	13.23
	HRED	25.06	11.69	6.26	3.71	13.65	2.84	13.04
	Our Model	**26.92**	**12.88**	**7.12**	**4.32**	**14.97**	**3.41**	**14.36**
Avg. Performance	Seq2Seq	25.25	11.99	6.54	3.92	13.87	3.09	13.34
	HRED	25.45	12.62	7.34	4.77	14.14	3.34	13.58
	Our Model	**26.93**	**13.57**	**8.00**	**5.21**	**15.13**	**3.78**	**14.54**

4.4 Results and Analysis

4.4.1 Automatic Evaluation

Here we evaluate the similarity of generated distractors with the ground truth. We employ BLEU (1-4) [59] and ROUGE (R1, R2, R-L) [45] scores to evaluate the similarity. BLEU evaluates average n-gram precision on a set of reference sentences, with a penalty for overly long sentences. ROUGE_1 and ROUGE_2 is the recall of unigrams and bigrams while ROUGE_L is the recall of longest common subsequences.

Table 4.2 shows the automatic evaluation results of all systems. Our model with static and dynamic attentions achieve the best performance across all metrics. We can observe a large performance gap between Seq2Seq and models with hierarchical architectures (HRED and our model), which reveals the hierarchical structure is useful for modeling the long sequential input. Another reason could be that some distractors can be generated only use information in several sentences, and sentence-level attentions (both static and dynamic) are useful to emphasize several sentences in the article. Moreover, our model with static attention achieves better performance than its ablation HRED, which shows the static attention can play the role of a soft gate to mask some irrelevant sentences and modulate the dynamic attention.

By comparing the three distractors generated by beam search with a predefined Jaccard distance, we find that the performance drops a little for the second and third distractors. The reason can be two-folds: 1) The second and third distractors have lower likelihood; 2) We set a Jaccard distance threshold as 0.5 to select

Table 4.3: Human evaluation results. Note that we allow annotators to choose more than one options if the generated outputs are accidentally the same or very semantically similar, therefore, the total number of selected options (552) is larger than the total number of annotated questions (540).

	Annotator 1	Annotator 2	Annotator 3	# Selected
Seq2Seq	31	35	30	96
HRED	33	40	35	108
Our Model	43	45	36	124
Human	75	70	79	224

the second and third distractors, thus they are forced to use some words different from those in the first distractor which is likely to be the best generation.

It is worth to mention that another automatic evaluation method can be applying a state-of-the-art reading comprehension pipeline for RACE to test its performance on our generated distractors. However, the current best performance of such reading comprehension pipeline is only 53.3% [90, 109, 99, 86], which means half questions in the dataset cannot be answered correctly. Therefore, we do not employ such reading comprehension pipeline to evaluate our generated distractors, instead we hire human annotators to conduct a reliable evaluation, given in the next section.

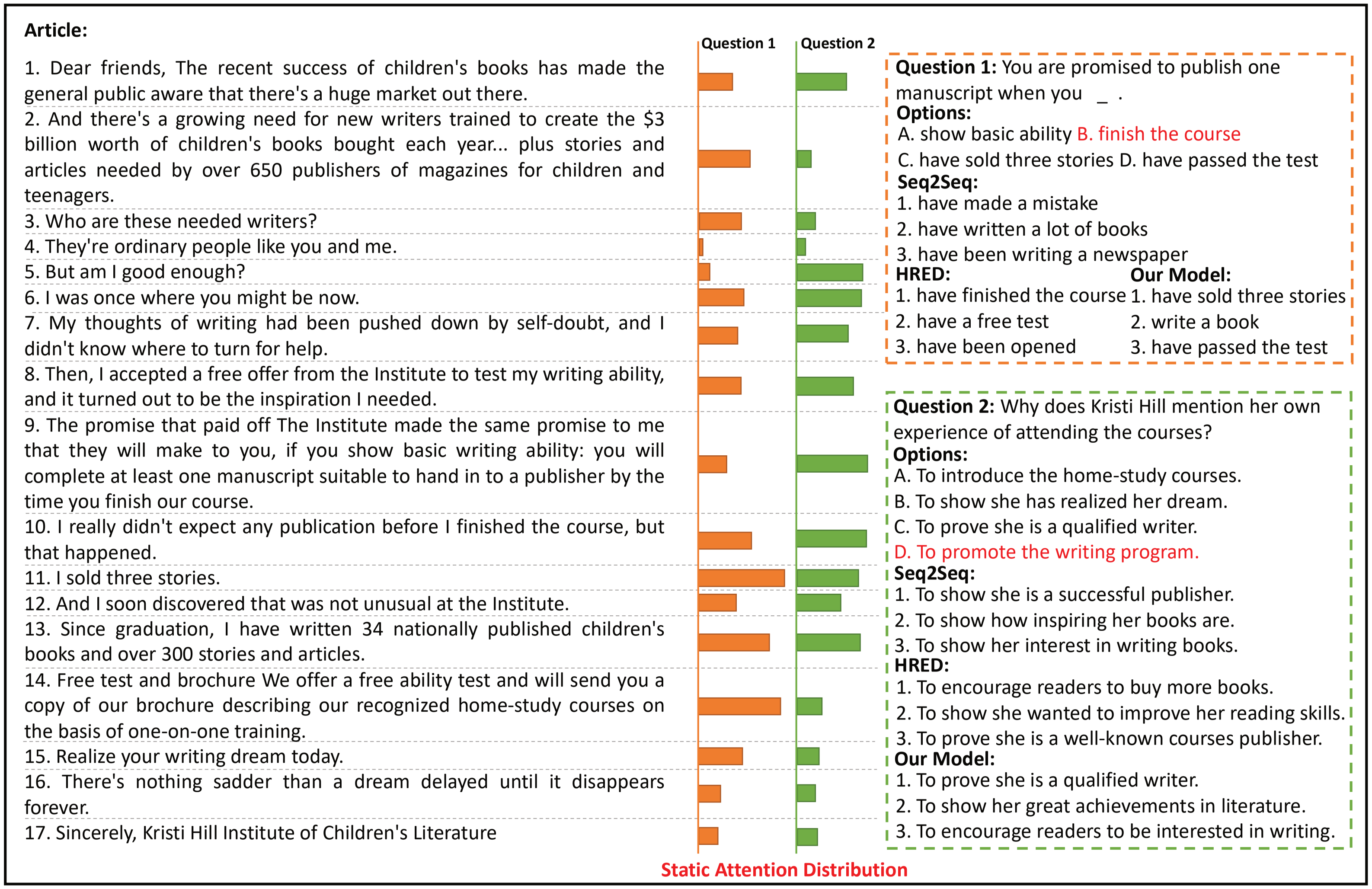
4.4.2 Human Evaluation

We conduct a human evaluation to investigate if the generated distractors can confuse the examinees in the real human test. We employ three annotators with good English background (at least holding a bachelor degree) to answer the MCQs with the generated distractors from different methods. Specifically, for each MCQ, we give 4 distractors as its options: One is a sample from the ground truth, the other three are generated by Seq2Seq, HRED, and our model respectively. Note that we did not give the correct answer option to the annotators, because the current human ceiling performance on RACE dataset is about 95% [35]. Thus, we need to do a huge amount of annotation for collecting enough questions that are answered wrongly. During the annotation, we told the annotators to select the most suitable option without considering whether there exists a correct option.

For comparison, we count how many times of individual pipelines (the ground truth and three compared methods) are successful in confusing the annotators, i.e. their distractors are selected as answers. We give each annotator 60 articles, and 3 questions per article. In total, we annotated 540 questions, and the results are given in Table 4.3. We find that the ground truth distractors (i.e. by "Human") have the strongest capability to confuse the annotators. Among the compared automatic methods, our model performs the best, while Seq2Seq performs the worst, which is a consistent conclusion as drawn from the previous section.

4.4.3 Case Study

In Figure 4.3, we present some sample distractors generated by human instructors, the Seq2Seq baseline, HRED and our model. To validate the effectiveness of the static attention, we show the static attention distributions over the sentences of the article for the two example questions. The correct options of the questions are marked in red.



Article:

1. Dear friends, The recent success of children's books has made the general public aware that there's a huge market out there.
2. And there's a growing need for new writers trained to create the $3 billion worth of children's books bought each year... plus stories and articles needed by over 650 publishers of magazines for children and teenagers.
3. Who are these needed writers?
4. They're ordinary people like you and me.
5. But am I good enough?
6. I was once where you might be now.
7. My thoughts of writing had been pushed down by self-doubt, and I didn't know where to turn for help.
8. Then, I accepted a free offer from the Institute to test my writing ability, and it turned out to be the inspiration I needed.
9. The promise that paid off The Institute made the same promise to me that they will make to you, if you show basic writing ability: you will complete at least one manuscript suitable to hand in to a publisher by the time you finish our course.
10. I really didn't expect any publication before I finished the course, but that happened.
11. I sold three stories.
12. And I soon discovered that was not unusual at the Institute.
13. Since graduation, I have written 34 nationally published children's books and over 300 stories and articles.
14. Free test and brochure We offer a free ability test and will send you a copy of our brochure describing our recognized home-study courses on the basis of one-on-one training.
15. Realize your writing dream today.
16. There's nothing sadder than a dream delayed until it disappears forever.
17. Sincerely, Kristi Hill Institute of Children's Literature

Question 1: You are promised to publish one manuscript when you _ .
Options:
A. show basic ability B. finish the course
C. have sold three stories D. have passed the test
Seq2Seq:
1. have made a mistake
2. have written a lot of books
3. have been writing a newspaper
HRED:
1. have finished the course
2. have a free test
3. have been opened
Our Model:
1. have sold three stories
2. write a book
3. have passed the test

Question 2: Why does Kristi Hill mention her own experience of attending the courses?
Options:
A. To introduce the home-study courses.
B. To show she has realized her dream.
C. To prove she is a qualified writer.
D. To promote the writing program.
Seq2Seq:
1. To show she is a successful publisher.
2. To show how inspiring her books are.
3. To show her interest in writing books.
HRED:
1. To encourage readers to buy more books.
2. To show she wanted to improve her reading skills.
3. To prove she is a well-known courses publisher.
Our Model:
1. To prove she is a qualified writer.
2. To show her great achievements in literature.
3. To encourage readers to be interested in writing.

Figure 4.3: Sample generated distractors. On the right, two example questions are given in dotted lines of yellow and green, and their corresponding static attention distributions are given in the middle by the bars of the corresponding colors.

Question 1 asks a detailed aspect in the article, which can be directly answered according to the 9th sentence. Since our static attention mechanism suppresses the sentences which contain the answer information, we can see the score for the 9th sentence is relatively smaller than others. The distractor outputs also justify our intuitions. Specifically, the first distractor by **HRED** is semantically identical to the correct option, thus it is not an appropriate distractor. With the help of the static attention, our model does not generate distractors like this. Another effect of the static attention is that it highlights the sentences that are relevant to the question, such as 11th, 13th, and 14th sentences, so that our model can generate better distractors. We can see the distractors generated by our model are semantically relevant to these highlighted sentences. Last but not least, we find that the distractors generated by Seq2Seq baseline either focus on some frequent words in the article such as *publish* and *write*, or contain some completely irrelevant words such as *mistake* and *newspaper*. HRED and our model do not have this problem, because the dynamic hierarchical attention can modulate the word-level attention distribution with the sentence-level attention.

By looking at Question 2, we can also find that the distractors generated by our system are more appropriate and relevant. Because Question 2 requires some inference, it is thus relevant to several sentences across the article. The static attention distribution yields the same conclusion. Specifically, the distribution shows that the 5th to 13th sentences are all relevant to the question, while the 14th sentence which is relevant to the answer option is suppressed. The generated distractors from our system are also semantically relevant to the 5th to 13th sentences.

4.5 Summary

In this chapter, we present a data-driven approach to generate distractors from multiple choice questions in reading comprehension from real examinations. We propose a hierarchical encoder-decoder framework with dynamic and static attention mechanisms to generate the context relevant distractors satisfying several constraints. We also prepare the first dataset for this new setting, and our model achieves the best performance in both automatic evaluation and human evaluation. For the future work, one interesting direction is to transform this one-to-many mapping problem into one-one mapping problem to better leverage the capability of the sequence-to-sequence framework. Another promising direction could be explicitly adding supervision signals to train the static attention. From the perspective of RACE-like reading comprehension tasks with multiple choice questions, although the performance of existing reading comprehension methods are still quite unsatisfactory, by introducing the distractor generation task, it might open another door for improving the performance, i.e. making adversarial approaches for solving this reading comprehension task possible.

□ **End of chapter.**

Chapter 5

Conversational Question Generation

In this chapter, We study the problem of generating interconnected questions in question-answering style conversations. Compared with previous works which generate questions based on a single sentence (or paragraph), this setting is different in two major aspects: (1) Questions are highly conversational. Almost half of them refer back to conversation history using coreferences. (2) In a coherent conversation, questions have smooth transitions between turns. We first give an introduction in Section 5.1. Then we define the task of conversational question generation in Section 5.2. In Section 5.3, we propose an end-to-end neural model with coreference alignment and conversation flow modeling to tackle this task. Section 5.4 and Section 5.5 describe our experimental setting and results. Finally, we give a summary of this chapter in Section 5.6.

Passage: Incumbent Democratic President Bill Clinton was ineligible to serve a third term due to term limitations in the 22nd Amendment of the Constitution, and Vice President Gore was able to secure the Democratic nomination with relative ease. Bush was seen as the early favorite for the Republican nomination and, despite a contentious primary battle with Senator John McCain and other candidates, secured the nomination by Super Tuesday. Bush chose ...

Q_1:	What political party is Clinton a member of?	A_1:	Democratic
Q_2:	What was he ineligible to serve?	A_2:	third term
Q_3:	Why?	A_3:	term limitations
Q_4:	Based on what amendment?	A_4:	22nd
Q_5:	Of what document?	A_5:	Constitution
Q_6:	Who was his vice president?	A_6:	Gore
Q_7:	Who was the early Republican favorite for the nomination?	A_7:	Bush
Q_8:	Who was the primary battle with?	A_8:	John McCain
Q_9:	What is his title?	A_9:	Senator
Q_{10}:	When did Bush secure the nomination by?	A_{10}:	Tuesday

Figure 5.1: An example for conversational question generation from a conversational question answering dataset CoQA [67]. Each turn contains a question Q_i and an answer A_i.

5.1 Introduction

Question Generation (QG) aims to create human-like questions from a range of inputs, such as natural language text [25], knowledge base [73] and image [58]. QG is helpful for the knowledge testing in education, i.e., the intelligence tutor system, where an instructor can actively ask questions to students given reading comprehension materials [25, 15]. Besides, raising good questions in a conversational can enhance the interactiveness and persistence of human-machine interactions [93].

Recent works on question generation for knowledge testing are mostly formalized as a standalone interaction [104, 76], while it is a more natural way for human beings to test knowledge or seek information through conversations involving a series of

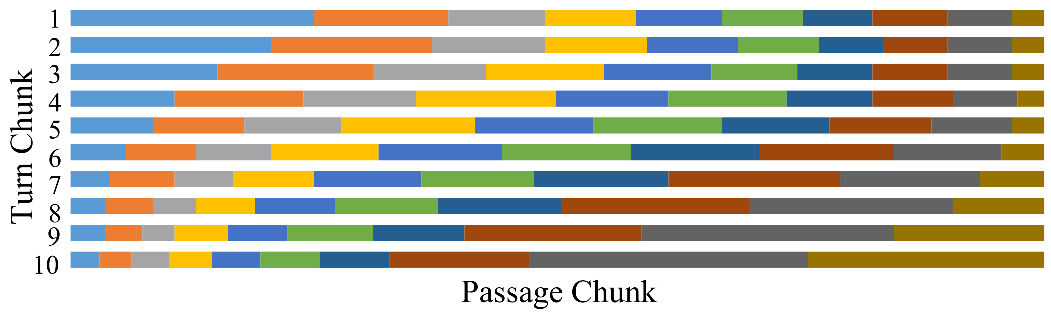


Figure 5.2: Passage chunks of interest for each turn chunks. Each row contains 10 bands distinguished by different colors. Each band represents a passage chunk. The width of a passage chunk indicates the concentration of conversation in that turn. The y-axis indicates turn chunk number. Same passage chunks share the same color across different turn chunks. *(Best viewed in color)*

interconnected questions [67]. Furthermore, the inability for virtual assistants to ask questions based on previous discussions often leads to unsatisfying user experiences. In this chapter, we consider a new setting called **C**onversational **Q**uestion **G**eneration (**CQG**). In this scenario, a system needs to ask a series of interconnected questions grounded in a passage through a question-answering style conversation. Table 5.1 provides an example under this scenario. In this dialogue, a questioner and an answerer chat about the above passage. Every question after the first turn is dependent on the conversation history.

Considering that the goal of the task is to generate interconnected questions in conversational question answering, CQG is challenging in a few aspects. Firstly, a model should learn to generate conversational interconnected questions depending on the conversation so far. As shown in Table 5.1, Q_3 is a single word 'Why?', which should be 'Why was he ineligible to serve a third term?' in a standalone interaction. Moreover, many questions in this conversation refer back to the conversation history using coreferences (e.g., Q_2, Q_6, Q_9), which is the nature of questions in a human conversation. Secondly, a coherent conversation must have smooth transitions between turns (each turn contains a question-answer pair). We expect the narrative structure of passages can influence the conversation flow of our interconnected questions. We further investigate this point by conducting an analysis on our experiment dataset CoQA [67]. We first split passages and turns of QA pairs into 10 uniform chunks and identify passage chunks of interest for each turn chunk. Figure 5.2 portrays the conversation flow between passage chunks and turn chunks. We see that in Figure 5.2, a question-answering style conversation usually starts focusing on the first few chunks in the passage and as the conversation advances, the focus shifts to the later passage chunks.

Previous works on question generation employ attentional sequence-to-sequence models on the crowd-sourced machine reading comprehension dataset SQuAD [66]. They mainly focus on generating questions based on a single sentence (or paragraph) and an answer phrase [15, 81, 105], while in our setting, our model needs to not only ask a question on the given passage (paragraph) but also make the questions conversational by considering the conversation history. Meanwhile, some researchers study question generation in dialogue systems to either achieve the correct answer through interactions [41] or enhance the interactiveness and persistence of conversations [93]. Although questions in our setting are conversational, our work is different from these because our conversations are grounded in the given passages rather than open-domain dialogues.

We propose a framework based on the attentional encoder-decoder model [51] to address this task. To generate conversational questions (first challenge), we propose a multi-source encoder to jointly encode the passage and the conversation so far. At each decoding timestep, our model can learn to focus more on the passage to generate content words or on the conversation history to make the question succinct. Furthermore, our coreference alignment modeling explicitly aligns coreferent mentions in conversation history (e.g. *Clinton* in Q_1 Table 5.1) with corresponding pronominal references in generated questions (e.g. *he* in Q_2), which makes generated questions interconnected to conversation history. The coreference alignment is implemented by adding extra supervision to bias the attention probabilities through a loss function. The loss function explicitly guides our model to resolve to the correct non-pronominal coreferent mentions in the attention distribution and generate the correct pronominal references in target questions. To make the conversations coherent (second challenge), we propose to model the conversation flow to transit focus inside the passage smoothly across turns. The conversation flow modeling achieves this goal via a flow embedding and a flow loss. The flow embedding conveys the correlations between number of turns and narrative structure of passages. The flow loss explicitly encourages our model to focus on sentences contain key information to generate the current turn question and ignore sentences questioned several turns ago.

In evaluations on a conversational question answering dataset CoQA [67], we find that our proposed framework outperforms several baselines in both automatic and human evaluations. Moreover, the coreference alignment can greatly improve the precision and recall of generated pronominal references. The conversation flow modeling can learn the smooth transition of conversation flow across turns.

5.2 Problem Setting

In this section, we define the Conversation Question Generation (CQG) task. Given a passage P, a conversation history $C_{i-1} = \{(Q_1, A_1), ..., (Q_{i-1},\ A_{i-1})\}$ and the aspect to ask (the current answer A_i), the task of CQG is to generate a question $\overline{Q_i}$ for the next turn:

$$\overline{Q}_i = \arg\max_{Q_i} \text{Prob}(Q_i | P, A_i, C_{i-1}), \tag{5.1}$$

in which the generated question should be as conversational as possible.

Note that we formalize this setting as an answer-aware QG problem [105], which assumes answer phrases are given before generating questions. Moreover, answer phrases are shown as text fragments in passages. Similar problems have been addressed in [14, 105, 81]. Our problem setting can also be generalized to the answer-ignorant case. Models can identify which answers to ask first by combining question-worthy phrases extraction methods [14, 91].

5.3 Model Description

As shown in Figure 5.3, our framework consists of four components: (1) multi-source encoder; (2) decoder with copy mechanism; (3) coreference alignment; (4) conversation flow modeling.

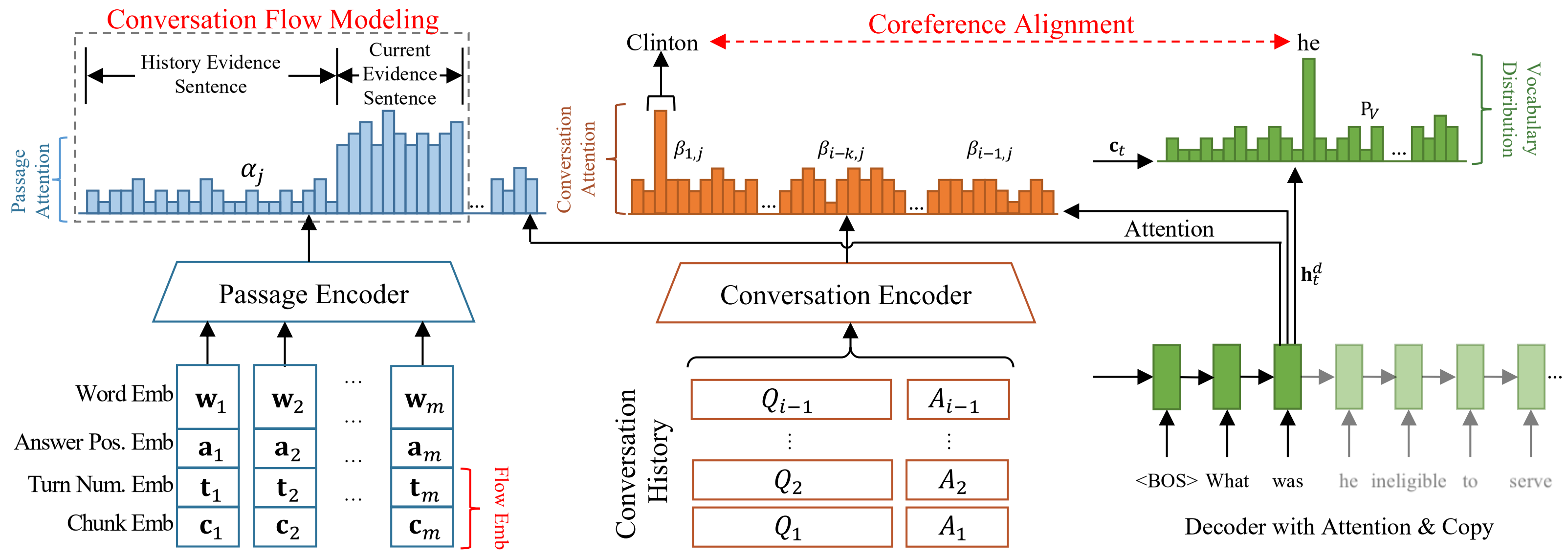


Figure 5.3: The framework of our proposed model. For clarity, we omit to plot the copy mechanism in the figure. *(Best viewed in color)*

5.3.1 Multi-Source Encoder

Since a conversational question is dependent on a certain aspect of the passage P and the conversation context C_{i-1} so far, we jointly encode information from two sources via a passage encoder and a conversation encoder.

Passage Encoder. The passage encoder is a bidirectional-LSTM (bi-LSTM) [27], which takes the concatenation of word embeddings $\mathbf{w}$ and answer position embeddings $\mathbf{a}$ as input $\mathbf{x}_i = [\mathbf{w}_i; \mathbf{a}_i]$. We denote the answer span using the typical BIO tagging scheme and map each token in the paragraph into the corresponding answer position embedding (i.e., `B_ANS`, `I_ANS`, `O`). Then the whole passage can be represented using the hidden states of the bi-LSTM encoder, i.e., $(\mathbf{h}_1^p, ..., \mathbf{h}_m^p)$, where m is the sequence length.

Conversation Encoder. The conversation history C_{i-1} is a sequence of question-answer pairs $\{(Q_1, A_1), ..., (Q_{i-1}, A_{i-1})\}$. We use segmenters $<q><a>$to concatenate each question answer pair (Q, A) into a sequence of tokens ($<q>$, $q_1, ..., q_m$; $<a>$, $a_1, ..., a_m$). We design a hierarchical structure to conduct conversation history modeling. We first employ a token level bi-LSTM to get contextualized representation of question-answer pairs $(\mathbf{h}_{i-k,1}^w, ..., \mathbf{h}_{i-k,m}^w)$, where $i - k$ is the turn number and $k \in [1, i)$. To model the dependencies across turns in the conversation history, we adopt a context level bi-LSTM to learn the contextual dependency $(\mathbf{h}_1^c, ..., \mathbf{h}_{i-1}^c)$ across different turns (denoted in the subscript $1, ..., i - 1$) of question-answer pairs.

5.3.2 Decoder with Attention & Copy

The decoder is another LSTM to predict the word probability distribution. At each decoding timestep t, it reads the word embedding $\mathbf{w}_t$ and the hidden state of previous timestep $\mathbf{h}^d_{t-1}$ to generate the current hidden state $\mathbf{h}^d_t = \text{LSTM}(\mathbf{w}_t, \mathbf{h}^d_{t-1})$.

To generate a conversational question grounded in the passage, the decoder itself should decide to focus more on passage hidden states $\mathbf{h}^p_j$ or the hidden states of conversation history $\mathbf{h}^w_{i-k,j}$ at each decoding timestep. Therefore, we flat token level conversation hidden states $\mathbf{h}^w_{i,j}$ and aggregate the passage hidden states $\mathbf{h}^p_j$ with token level conversation hidden states $\mathbf{h}^w_{i,j}$ into a unified memory: $(\mathbf{h}^p_1, ..., \mathbf{h}^p_m; \mathbf{h}^w_{1,1}, ..., \mathbf{h}^w_{1,m}; ... ; \mathbf{h}^w_{i-1,1}, ..., \mathbf{h}^w_{i-1,m})$, where $\mathbf{h}^w_{i,j}$ denotes the j-th token of the i-th turn in token level conversation hidden states. Then we attend the unified memory with the standard attention mechanism [51] for the passage attention $(\alpha_1, ..., \alpha_m)$ and the hierarchical attention mechanism for the conversation attention $(\beta_{1,1}, ..., \beta_{1,m}; ...; \beta_{i-1,1}, ..., \beta_{i-1,m})$:

$$e^p_j = \mathbf{h}^{p\top}_j \mathbf{W}_p \mathbf{h}^d_t, \tag{5.2}$$

$$e^w_{i-k,j} = \mathbf{h}^w_{i-k,j}{}^\top \mathbf{W}_w \mathbf{h}^d_t, \tag{5.3}$$

$$e^c_{i-k} = \mathbf{h}^c_{i-k}{}^\top \mathbf{W}_c \mathbf{h}^d_t, \tag{5.4}$$

$$\alpha_j = \frac{e^p_j}{e_{\text{total}}}, \quad \beta_{i-k,j} = \frac{e^w_{i-k,j} * e^c_{i-k}}{e_{\text{total}}}, \tag{5.5}$$

where $e_{\text{total}} = \Sigma_j e^p_j + \Sigma_{k,j} e^w_{i-k,j} * e^c_{i-k}$ and $\mathbf{W}_p$, $\mathbf{W}_w$, $\mathbf{W}_c$ are learnable weights.

Finally, we derive the context vector $\mathbf{c}_t$ and the final vocabulary distribution P_V:

$$\mathbf{c}_t = \Sigma_j \alpha_j \mathbf{h}^p_j + \Sigma_{j,k} \beta_{i-k,j} \mathbf{h}^w_{i-k,j},$$
$$\text{P}_V = \text{softmax}(\mathbf{W}_v(\tanh(\mathbf{W}_a[\mathbf{h}^d_t; \mathbf{c}_t]) + \mathbf{b}_v),$$

where $\mathbf{W}_v$, $\mathbf{W}_a$ are learnable weights. Please refer to [71] for more details on the copy mechanism.

5.3.3 Coreference Alignment

Using coreferences to refer back is an essential property of conversational questions. Almost half of the questions contains explicit coreference markers such as *he*, *she*, *it* in CoQA [67]. Therefore, we propose the coreference alignment to enable our model such ability. Take Q_2 in Table 5.1 as an example, traditional question generation system can only generate question like "What was *Clinton* ineligible to serve?", while our system with coreference alignment can align the name "*Clinton*" to its pronominal reference "*he*" and generate a more conversational question "What was *he* ineligible to serve?".

The coreference alignment modeling tells the decoder to look at the correct non-pronominal coreferent mention in the conversation attention distribution to produce the pronominal reference word. We achieve this via two stages. In the preprocessing stage, given the conversation history C_{i-1} and the question Q_i which has a pronominal reference (e.g., *he* for Q_2 in Table 5.1), we first run a coreference resolution system [9] to find its coreferent mention $(w_1^c, ...w_m^c)$ (e.g. *Clinton*) in the conversation history C_{i-1}, where the superscript c denotes tokens identified as the coreferent mention. During training, we introduce a novel loss function built on the conversation attention of coreferent mentions β_i^c and the output word probability of its pronominal reference word $p_{\text{coref}} \in \mathrm{P}_V$. As shown in Figure 5.3, when our model need to refer back to the coreferent mention, we ask the model focus correctly on the antecedent (e.g. *Clinton*) and maximize the probability of its pronominal reference (e.g. *he*) p_{coref} in the output vocabulary distribution P_V,

$$\mathcal{L}_{\text{coref}} = -(\lambda_1 \log \frac{\Sigma_j \beta_j^c}{\Sigma_{k,j} \beta_{i-k,j}} + \lambda_2 \log p_{\text{coref}}) * s_c,$$

where λ_1, λ_2 are hyperparameters, s_c is the confidence score between the non-pronominal coreferent mention and the pronoun obtained during the pre-processing stage.

5.3.4 Conversation Flow Modeling

Another key challenge in CQG is that a coherent conversation must have smooth transitions between turns. As illustrated in Figure 5.2, we find that as the conversations go on, most of the questioners transit their focus from the beginning of passages to the end. Following this direction, we model the conversation flow to learn smooth transitions across turns of the conversation.

Flow Embedding. As shown in Figure 5.3, we feed our model with the current turn number indicator in the conversation and the relative position for each token in the passage, which, intuitively, are useful for modeling the conversation flow. We achieve this goal via two additional embeddings. The turn number embedding is a learned lookup table $[\mathbf{t}_1, ..., \mathbf{t}_n]$ to map the turn number i into its feature embedding space, where n is the maximum turn we consider. For encoding the relative position of each token, we split the passage into L uniform chunks. Each token in the passage is mapped to its corresponding chunk embedding $[\mathbf{c}_1, ..., \mathbf{c}_L]$. The final input to the passage encoder is the concatenation of word embedding, answer position embedding (introduced in Section 5.3.1) and these two additional embeddings: $\mathbf{x}_i = [\mathbf{w}_i; \mathbf{a}_i; \mathbf{t}_i; \mathbf{c}_i]$.

We further add a gated self-attention modeling mechanism [105] in the passage encoder. Motivating our use of self-attention we consider two desiderata. One is self-attention with answer position embedding can aggregate answer-relevant information from the whole passage for question generation. Another is we want to learn the latent alignment between the turn number embedding and the chunk embedding for better modeling the conversation flow. We first match the rich-feature enhanced passage representation $\mathbf{H}^p = [\mathbf{h}^p_1; ...; \mathbf{h}^p_m]$ with itself $\mathbf{h}^p_j$ to compute the self-matching representation $\mathbf{u}^p_j$, and then combine it with the original representation $\mathbf{h}^p_j$:

$$\mathbf{a}^p_j = \text{softmax}(\mathbf{H}^{p\top}\mathbf{W}_s\mathbf{h}^p_j),\ \mathbf{u}^p_j = \mathbf{H}^p\mathbf{a}^p_j \tag{5.6}$$

$$\mathbf{f}^p_j = \tanh(\mathbf{W}_f[\mathbf{h}^p_j; \mathbf{u}^p_j]), \tag{5.7}$$

The final representation $\tilde{\mathbf{h}}^p_j$ is derived via a gated summation through a learnable gate vector $\mathbf{g}^p_j$,

$$\mathbf{g}^p_t = \text{sigmoid}(\mathbf{W}_g[\mathbf{h}^p_j; \mathbf{u}^p_j]) \tag{5.8}$$

$$\tilde{\mathbf{h}}^p_j = \mathbf{g}^p_t \odot \mathbf{f}^p_j + (1 - \mathbf{g}^p_t) \odot \mathbf{h}^p_j \tag{5.9}$$

where $\mathbf{W}_s$, $\mathbf{W}_f$, $\mathbf{W}_g$ are learnable weights, $\odot$ is the element-wise multiplication. Self matching enhanced representation $\tilde{\mathbf{h}}^p_j$ takes the place of the passage representation $\mathbf{h}^p_j$ for calculating the passage attention.

Flow Loss. In Section 5.3.1, our answer position embedding can help model the conversation flow by showing the position of answer fragments inside the passage. However, it is still helpful to tell the model explicitly which sentences around the answer are of high informativity to generate the current turn question. The flow loss is designed to help our model to locate the evidence sentences correctly. Firstly, we define two kinds of sentences in the passage. If a sentence is informative to the current question, we call it `Current Evidence Sentence (CES)`. If a sentence is informative to questions in the conversation history and irrelevant to the current question, we call it `History Evidence Sentence (HES)`. Then our model is taught to focus on current evidence sentences and ignore the history evidence sentences in the passage attention α_j via the following flow loss:

$$\mathcal{L}_{\text{flow}} = -\lambda_3 \log \frac{\Sigma_{j:w_j \in \texttt{CES}} \alpha_j}{\Sigma_j \alpha_j} + \lambda_4 \frac{\Sigma_{j:w_j \in \texttt{HES}} \alpha_j}{\Sigma_j \alpha_j}$$

where λ_3, λ_4 are hyperparameters, and $w_j \in \texttt{CES/HES}$ indicates the token w_j is inside the sentence with a `CES/HES` label.

5.3.5 Joint Training

Considering all the aforementioned components, we define a joint loss function as:

$$\mathcal{L} = \mathcal{L}_{\text{nll}} + \mathcal{L}_{\text{coref}} + \mathcal{L}_{\text{flow}}, \tag{5.10}$$

in which $\mathcal{L}_{\text{nll}} = -\log \text{Prob}(Q_i|P, A_i, C_{i-1})$ is the the negative log-likelihood loss in the sequence to sequence learning [82].

5.4 Experiments

5.4.1 Dataset Preparation

We conduct experiments on the CoQA dataset [67]. It is a large-scale conversational question answering dataset for measuring the ability of machines to participate in a question-answering style conversation. The authors employ Amazon Mechanical Turk to collect 8k conversations with 127k QA pairs. Specifically, they pair two crowd-workers: a questioner and an answerer to chat about a passage. The answerers are asked to firstly highlight extractive spans in the passage as rationales and then write the free-form answers. We first extract each data sample as a quadruple of passage, question, answer and conversation history (previous n turns of QA pairs) from CoQA. Then we filter out QA pairs with *yes*, *no* or *unknown* as answers (28.7% of total QA pairs) because there is too little information to generate the question to the point. Finally, we randomly split the dataset into a training set (80%, 66298 samples), a validation set (10%, 8409 samples) and a testing set (10%, 8360 samples). The average passage, question and answer lengths are 332.9, 6.3 and 3.2 tokens respectively.

5.4.2 Implementation Details

Locating Extractive Answer Spans. As studied by [101], abstractive answers in CoQA are mostly small modifications to spans occurring in the context. The maximum achievable performance by a model that predicts spans from the context is 97.8 F1 score. Therefore, we find the extractive spans from the passage which have the maximum F1 score with answers and treat them as answers for our answer position embedding.

Number of Turns in Conversation History. [67] find that in CoQA dataset, most questions in a conversation have a limited dependency within a bound of two turns. Therefore, we choose the number of history turns as $n = 3$ to ensure the target questions have enough conversation history information to generate and avoid introducing too much noise from all turns of QA pairs.

Labeling Evidence Sentences. As mentioned in Section 5.4.1, the crowd-workers label the extractive spans in the passage as rationales for actual answers. We treat sentences containing the rationale as `Current Evidence Sentence`.

Model Settings. We employ the teacher-forcing training, and in the generating stage, we set the maximum length for output sequence as 15 and block unigram repeated token, the beam size k is set to 5. All hyperparameters and models are selected on the validation set and the results are reported on the test set.

5.4.3 Baselines and Ablations

We compare with the state-of-the-art baselines and conduct ablations as follows: **PGNet** is the pointer-generator network [71]. We concatenate the passage P, the conversation history C_{i-1} and the current answer A_i as a sequence for the input. **NQG** [14] is similar to the previous one but it takes current answer features concatenated with the word embeddings during encoding. **MSNet** is our base model **M**ulti-**S**ource encoder decoder network (Section 5.3.1 & 5.3.2). **CorefNet** is our proposed **Coref**erence alignment model (Section 5.3.3). **FlowNet** is our proposed conversation **Flow** model (Section 5.3.4). **CFNet** is the model with both the **C**oreference alignment and the conversation **F**low modeling.

Table 5.1: Main results of baselines and our models. *t*-test is conducted between our CFNet and baselines/ablations. (<u>underline</u>: *p-value* <0.05, *: *p-value* <0.01).

	B1	B2	B3	R-L
PGNet	28.84*	13.74*	8.16*	39.18*
NQG	35.56*	21.14*	14.84*	45.58*
MSNet	36.27*	21.92*	15.51*	46.01*
CorefNet	<u>36.89</u>	<u>22.28</u>	<u>15.77</u>	<u>46.53</u>
FlowNet	<u>36.87</u>	22.49	15.98	46.64
CFNet	**37.38**	**22.81**	**16.25**	**46.90**

5.5 Results and Analysis

5.5.1 Main Results

Since the average length of questions is 6.3 tokens only, we employ BLEU (1-3) [59] and ROUGE-L (R-L) [45] scores to evaluate n-gram similarity between the generated questions with the ground truth. We evaluate baselines and our models by predicting the current question given a passage, the current answer, and the ground truth conversation history.

Table 5.1 shows the main results, and we have the following observations:

- NQG outperforms PGNet by a large margin. The improvement shows that the answer position embedding [108] is helpful for asking questions to the point.
- Our base model MSNet outperforms NQG, which reveals that the hierarchical encoding and the hierarchical attention to conversation history can model the dependency across different turns in conversations.
- Both our CorefNet and FlowNet outperform our base model.

Table 5.2: Evaluation results on the coreference test set. Precision (P), Recall (R) and F-score (F) of predicted pronouns are also reported. Significant tests with t-test are conducted between CorefNet and models without the coreference alignment. (underline: *p-value* <0.05, *: *p-value* <0.01).

	B1	B2	B3	R-L	P	R	F
PGNet	27.66*	13.82*	8.96*	38.40*	26.87*	25.17*	25.68*
NQG	34.75*	21.52*	15.96*	45.04*	34.46*	32.97*	33.25*
MSNet	36.31*	22.92	17.07	45.97*	35.34*	33.80*	34.07*
CorefNet	**37.51**	**24.14**	**18.44**	**47.45**	**42.09**	**40.35**	**40.64**

We will analyze the effectiveness of our coreference alignment and conversation flow modeling in the following two sections respectively.

- Our CFNet is significantly better than two baselines (PGNet, NQG), our MSNet, and our CorefNet. However, the difference between our CFNet and our FlowNet is not significant. This is because the conversation flow modeling improves all test samples while the coreference alignment contributes only to questions containing pronominal references.

5.5.2 Coreference Alignment Analysis

As we discussed in Section 5.3.3, it is the nature of conversational questions to use coreferences to refer back. In order to demonstrate the effectiveness of the proposed coreference alignment, we evaluate models on a subset of the test set called coreference set. Each sample in the coreference set requires a pronoun resolution between the conversation history and the current question (e.g., Q_2, Q_6, Q_9 in Table 5.1). In additional to the BLEU(1-3) and ROUGE-L metrics, we also calculate the Precision (P), Recall (R) and F-score (F) of pronouns in the

Passage: … however , mccain has a very different life story . he grew up in a navy family and was a pilot during the vietnam war in the 1960s …

Conversation History:

<q>	what	war	was	mccain	in	?
0.0000	0.0001	0.0049	0.0138	0.7710	0.0055	0.0069
<a>	vietnam	war				
0.0000	0.0140	0.0095				
<q>	was	he	in	the	army	?
0.0000	0.0045	0.1303	0.0005	0.0139	0.0001	0.0250
<a>	no					
0.0000	0.0000					

Question (Human): what was his job ?

Question (Our Model): what was his job ?

Passage: … incumbent democratic president bill clinton was ineligible to serve a third term due to term limitations in the 22nd amendment of the constitution …

Conversation History:

<q>	what	political	party	is	clinton	a
0.0000	0.0000	0.0002	0.0063	0.0045	0.9260	0.0430
member	of	?	<a>	democratic		
0.0008	0.0006	0.0026	0.0000	0.0160		

Question (Human): what was he ineligible to serve ?

Question (Our Model): what was he ineligible for ?

Figure 5.4: Examples for the coreference alignment model. We show the attention probability (renormalize to 1) when the CorefNet predicts a pronoun (red color in Question). The current answers are underlined in the passages. *(Best viewed in color)*

generated questions with regard to pronouns in the ground truth questions.

The results are depicted in Table 5.2. With the help of the coreference alignment, CorefNet significantly improves the precision, recall, and f-score of the predicted pronouns.

Moreover, the performance on n-gram overlapping metrics is also boosted. To gain more insights into how the coreference alignment model influence the generation process, in Figure 5.4, we visualize the conversation attention distribution β_j at the timestep the model predicts a pronoun. The conversation history distribution β_j is renormalized to $\Sigma_j \beta_j = 1$. All two examples show that our model put the highest attention probability on the coreferent mentions (i.e. *McCain/Clinton*) when it generates the pronominal references (*his/he*). We can conclude that our coreference alignment model can align correct coreferent mentions to generate corresponding pronouns.

5.5.3 Conversation Flow Modeling Analysis

As discussed in Section 5.3.4, a coherent conversation should have smooth transitions between turns, and we design our model to follow the narrative structure of the passage. Figure 5.5 shows an example illustrating the transition of passage attention distribution a_j (normalize to 1) during first 11 turns of a conversation. We see that the model transits its focus smoothly across the first 11 turns from the first sentence in the passage to later parts. Sometimes the model drills down with two questions for the same sentence such as turn 2 & 3, 4 & 5 and 10 & 11.

To quantitatively validate the effectiveness of our conversation flow modeling, we study the alignment between passage attention α_j and sentences of interest in the passage. Ideally, a successful model should focus on sentences of interest (i.e., Current Evidence Sentence) and ignore sentences questioned several turns ago (i.e., History Evidence Sentence). We validate this intuition by calculating $\Sigma_{j:w_j \in \texttt{CES}} \alpha_j$ and $\Sigma_{j:w_j \in \texttt{HES}} \alpha_j$ for

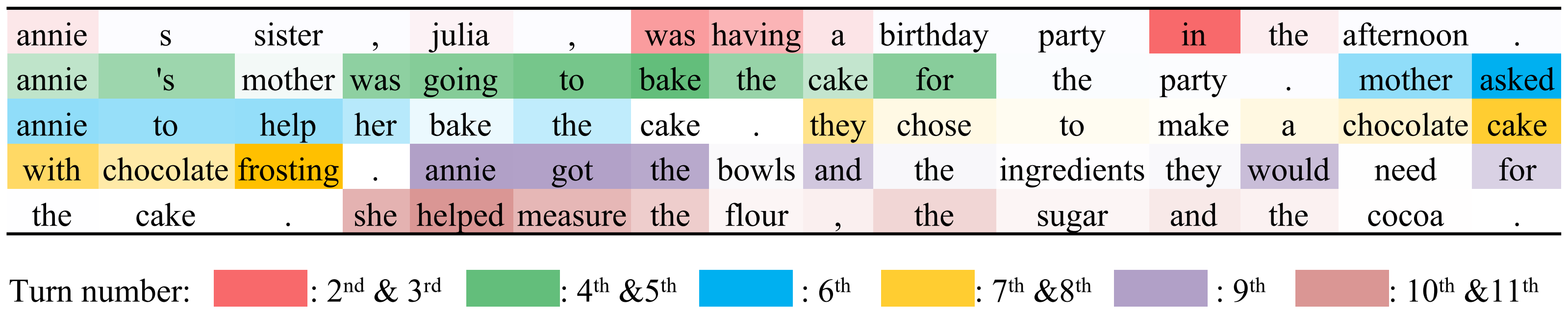


Figure 5.5: The transition of passage attention distribution between turns. Different colors are correspond to different turns. To show attention probability of different turns in one place, we only draw attention probability α_j >0.1 here. If two turns focus on the same sentence, we average the attention probability between them. *(Best viewed in color)*

all examples in test set. Results show that $\Sigma_{j:w_j \in \texttt{CES}} \alpha_j$ and $\Sigma_{j:w_j \in \texttt{HES}} \alpha_j$ for our model with conversation flow modeling are 0.9966 and 0.0010 on average, which demonstrates that our conversation flow modeling can locate the current evidence sentences precisely and ignore the history evidence sentence. For the model without the flow modeling (CorefNet), $\Sigma_{j:w_j \in \texttt{CES}} \alpha_j = 0.4093$, $\Sigma_{j:w_j \in \texttt{HES}} \alpha_j = 0.1778$, which proves our intuition in Section 5.3.4 that the answer position embedding cannot have comparable effects on the conversation flow modeling.

5.5.4 Human Evaluation

We randomly sample 93 questions with the associated passage and conversation history to conduct human evaluation. We hire 5 workers to evaluate the questions generated by PGNet, MSNet, and our CFNet. All models are evaluated in terms of following 3 metrics: "Grammaticality", "Answerability" and "Interconnectedness". "Grammaticality" measures the grammatical correctness and fluency of the generated questions. "Answerability" evaluates whether the generated question can be answered by the current answer. "Interconnectedness" measures whether the generated questions are *conversational* or not. If a question refers back to the conversation history using coreference or is dependent on the conversation history such as incomplete questions 'Why?', 'Of what?', we define it as a *conversational* question. All metrics are rated on a 1-3 scale (3 for the best).

The results are shown in Table 5.3. All models achieve high scores on "Grammaticality", owing to the strong language modeling capability of neural models. MSNet and our CFNet

Table 5.3: Manual evaluation results. All metrics are rated on a 1-3 scale (3 for the best). Two-tailed *t*-test results are shown for our CFNet compared to PGNet/MSNet. * indicates *p-value* <0.01.

	Grammaticality	Answerability	Interconnectedness
PGNet	2.74	1.39	1.59
MSNet	2.85	2.39	1.74
CFNet	2.89	**2.74***	**2.67***

perform well on "Answerability" while PGNet does not. This demonstrates our base model MSNet and our CFNet can ask questions to the point. Finally, our CFNet outperforms the other two models in terms of "Interconnectedness" by a large gap, which proves that the proposed coreference alignment and conversation flow modeling can effectively make questions *conversational*.

5.6 Summary

In this chapter, we study the problem of question-answering style Conversational Question Generation (CQG), which has never been investigated before. We propose an end-to-end neural model with coreference alignment and conversation flow modeling to solve this problem. Experiments show that our proposed framework achieves the best performance in automatic and human evaluations.

□ **End of chapter.**

Chapter 6

Explicit Memory Tracker for Conversational Machine Reading

The goal of conversational machine reading is to answer user questions given a knowledge base text which may require asking clarification questions. Existing approaches are limited in their decision making due to struggles in extracting question-related rules and reasoning about them. In this chapter, we present a new framework of conversational machine reading that comprises a novel **E**xplicit **M**emory **T**racker (EMT) to track whether conditions listed in the rule text have already been satisfied to make a decision. We first give an introduction in Section 6.1. Then we describe our proposed EMT model in Section 6.2. Section 6.3 describe our experimental setting and results. Finally, we give a summary of this chapter in Section 6.4.

6.1 Introduction

In conversational machine reading (CMR), machines can take the initiative to ask users questions that help to solve their problems, instead of jumping into a conclusion hurriedly [69]. In this case, machines need to understand the knowledge base (KB) text, evaluate and keep track of the user scenario, ask clarification questions, and then make a final decision. This interactive behavior between users and machines has gained more attention recently because in practice users are unaware of the KB text, thus they cannot provide all the information needed in a single turn.

For instance, consider the example in Figure 6.1 taken from the ShARC dataset for CMR [69]. A user posts her scenario and asks a question on whether her employer can take money from her final pay. Since she does not know the relevant rule text, the provided scenario and the initial question(s) from her are often too underspecified for a machine to make a certain decision. Therefore, a machine has to read the rule text and ask a series of clarification questions until it can conclude the conversation with a certain answer.

Most existing approaches [107, 89] formalize the CMR problem into two sub-tasks. The first is to make a decision among `Yes`, `No`, `Irrelevant`, and `Inquire` at each dialog turn given a rule text, a user scenario, an initial question and the current dialog history. If one of `Yes`, `No`, or `Irrelevant` is selected, it implies that a final decision (`Yes`/`No`) can be made in response to the user's initial question, or stating the user's initial question is unanswerable (`Irrelevant`) according to the rule text. If the decision at the current turn is `Inquire`, it will then trigger the

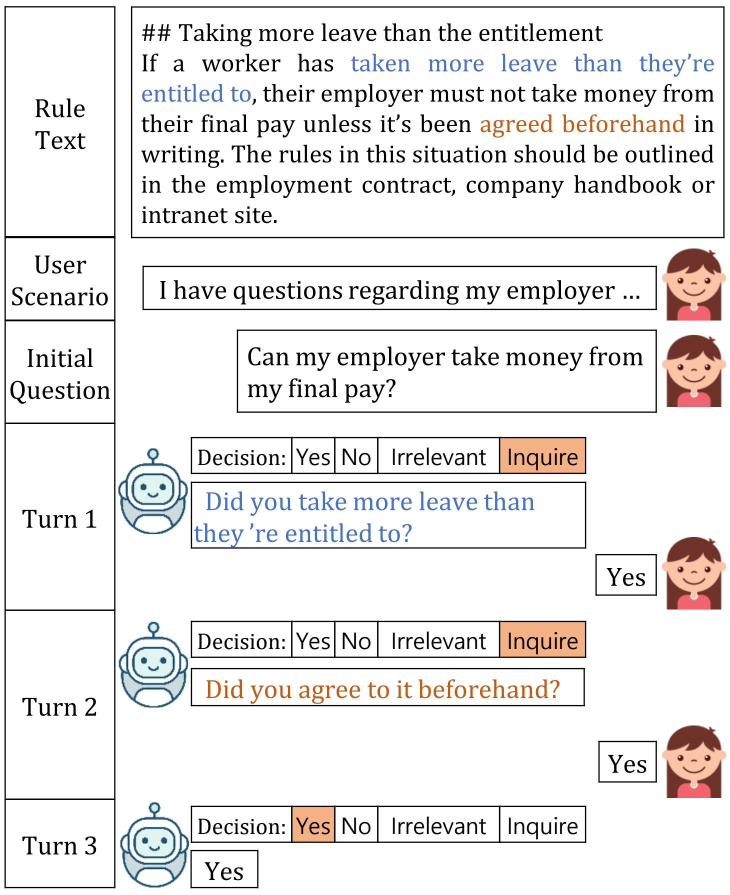

Rule Text
## Taking more leave than the entitlement
If a worker has taken more leave than they're entitled to, their employer must not take money from their final pay unless it's been agreed beforehand in writing. The rules in this situation should be outlined in the employment contract, company handbook or intranet site.
User Scenario
I have questions regarding my employer ...
Initial Question
Can my employer take money from my final pay?
Turn 1
Decision: Yes No Irrelevant Inquire
Did you take more leave than they 're entitled to?
Yes
Turn 2
Decision: Yes No Irrelevant Inquire
Did you agree to it beforehand?
Yes
Turn 3
Decision: Yes No Irrelevant Inquire
Yes

Figure 6.1: Example of Conversational Machine Reading tasks from the ShARC dataset [69]. At each turn, given the rule text, a user scenario, an initial user question, and previous interactions, a machine can give a certain final answer such as `Yes` or `No` to the initial question. If the machine cannot give a certain answer because of missing information from the user, it will ask a clarification question to fill in the information gap. Clarification questions and their corresponding rules are marked in the same colors.

second task for follow-up question generation, which extracts an underspecified rule span from the rule text and generates a follow-up question accordingly.

However, there are two main drawbacks to the existing methods. First, with respect to the reasoning of the rule text, existing methods do not explicitly track whether a condition listed in the rule has already been satisfied as the conversation flows so that it can make a better decision. Second, with respect to the extraction of question-related rules, it is difficult in the current approach to extract the most relevant text span to generate the next question. For example, the state-of-the-art E^3 model [107] has only 60.6% F1 for question-related span extraction.

To address these issues, we propose a new framework of conversational machine reading with a novel **E**xplicit **M**emory **T**racker (EMT), which explicitly tracks each rule sentence to make decisions and generate follow-up questions. Specifically, EMT first segments the rule text into several rule sentences and allocates them into its memory. Then the initial question, user scenario, and dialog history are fed into EMT sequentially to update each memory module separately. At each dialog turn, EMT predicts the entailment states (satisfaction or not) for every rule sentence, and makes a decision based on the current memory status. If the decision is `Inquire`, EMT extracts a rule span to generate a follow-up question by adopting a coarse-to-fine reasoning strategy (i.e., weighting token-level span distributions with its sentence-level entailment scores). Compared to previous methods which only consider entailment-oriented reasoning for decision making or follow-up question generation, EMT utilizes its updated memory modules to reason out these two tasks in a unified manner.

We compare EMT with the existing approaches on the ShARC dataset [69]. Our results show that explicitly tracking rules with external memories boosts both the decision accuracy and the quality of generated follow-up questions. In particular, EMT outperforms the previous best model E^3 by 1.3 in macro-averaged decision accuracy and 10.8 in BLEU4 for follow-up question generation. In addition to the performance improvement, EMT yields interpretability by explicitly tracking rules, which is visualized to show the entailment-oriented reasoning process of our model.

6.2 Method

As illustrated in Figure 6.2, our proposed method consists of the following four main modules.

(1) The *Encoding* module uses BERT [11] to encode the concatenation of the rule text, initial question, scenario and dialog history into contextualized representations.

(2) The *Explicit Memory Tracking* module sequentially reads the initial question, user scenario, multi-turn dialog history, and updates the entailment state of each rule sentence.

(3) The *Decision Making* module does entailment-oriented reasoning based on the updated states of rule sentences and makes a decision among `Yes`, `No`, `Irrelevant`, and `Inquire`.

(4) If the decision is `Inquire`, the *Question Generation* module is activated, which reuses the updated states of rule sentences to identify the underspecified rule sentence and

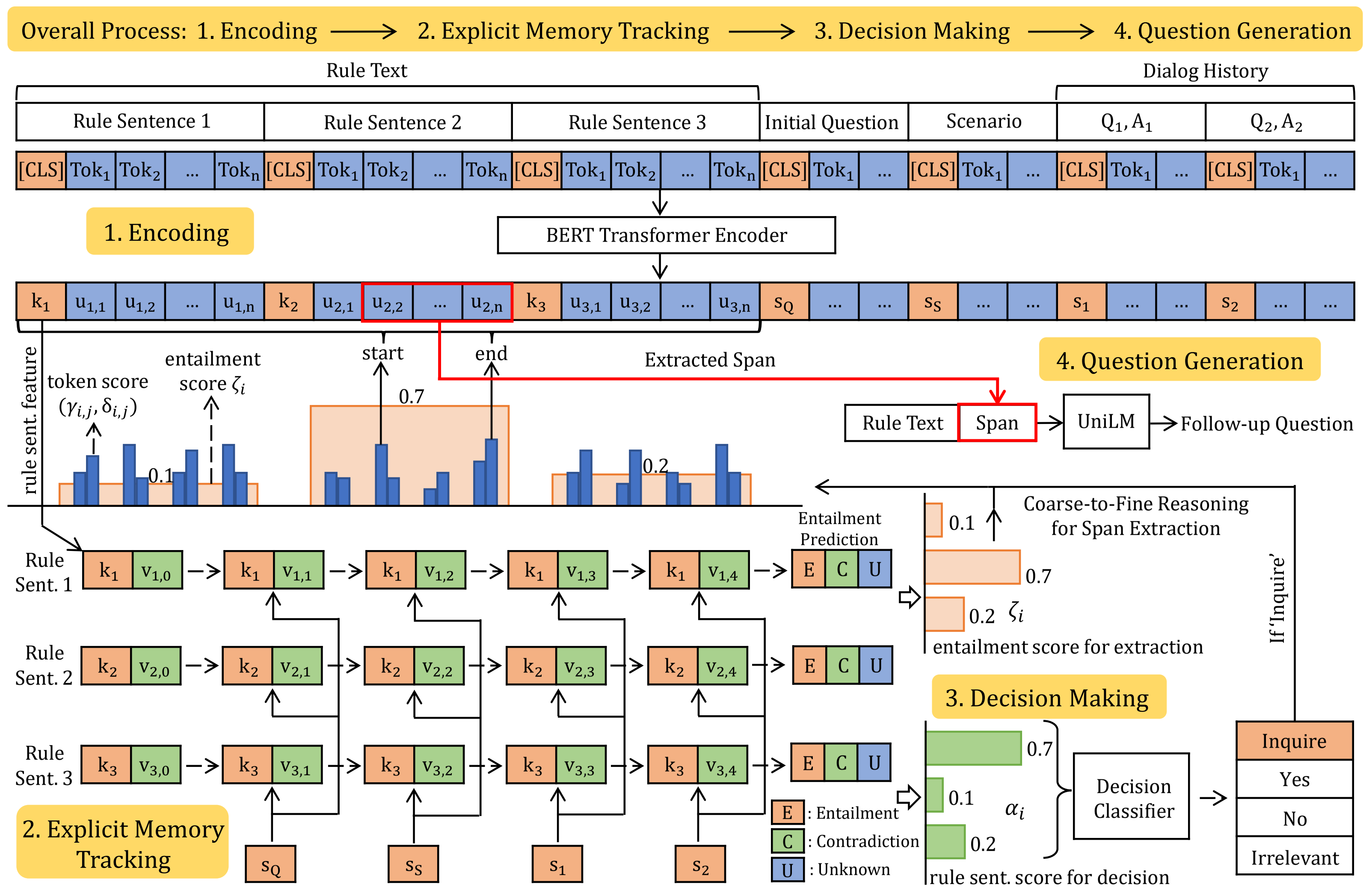

Overall Process: 1. Encoding → 2. Explicit Memory Tracking → 3. Decision Making → 4. Question Generation
Rule Text
Dialog History
Rule Sentence 1
Rule Sentence 2
Rule Sentence 3
Initial Question
Scenario
Q_1, A_1
Q_2, A_2
[CLS] Tok_1 Tok_2 ... Tok_n
1. Encoding
BERT Transformer Encoder
k_1 $u_{1,1}$ $u_{1,2}$ ... $u_{1,n}$ k_2 $u_{2,1}$ $u_{2,2}$ ... $u_{2,n}$ k_3 $u_{3,1}$ $u_{3,2}$ ... $u_{3,n}$ s_Q ... s_S ... s_1 ... s_2 ...
rule sent. feature
token score $(\gamma_{i,j}, \delta_{i,j})$
entailment score ζ_i
start
end
Extracted Span
0.1
0.7
0.2
4. Question Generation
Rule Text
Span
UniLM
Follow-up Question
Coarse-to-Fine Reasoning for Span Extraction
If 'Inquire'
Entailment Prediction
Rule Sent. 1
Rule Sent. 2
Rule Sent. 3
k_1 $v_{1,0}$ $v_{1,1}$ $v_{1,2}$ $v_{1,3}$ $v_{1,4}$
k_2 $v_{2,0}$ $v_{2,1}$ $v_{2,2}$ $v_{2,3}$ $v_{2,4}$
k_3 $v_{3,0}$ $v_{3,1}$ $v_{3,2}$ $v_{3,3}$ $v_{3,4}$
s_Q s_S s_1 s_2
E C U
2. Explicit Memory Tracking
E : Entailment
C : Contradiction
U : Unknown
ζ_i
entailment score for extraction
3. Decision Making
α_i
rule sent. score for decision
Decision Classifier
Inquire
Yes
No
Irrelevant

Figure 6.2: The Explicit Memory Tracker with Coarse-to-Fine Reasoning for Conversational Machine Reading (CMR). The CMR process includes (1) BERT encoding, (2) Explicit Memory Tracking for entailment state of each rule sentence, (3) Decision Making on updated entailment states of all rule sentences, (4) Question Generation via span extraction with coarse-to-fine reasoning and question rephrasing of the extracted span. *(Best viewed in color)*

extract the most informative span within it in a coarse-to-fine manner. Then it rephrases the extracted span into a well-formed follow-up question.

6.2.1 Encoding

Let x_R, x_Q, x_S, $[x_{H,1}, x_{H,2}, ..., x_{H,P}]$ denote the input of rule text, initial question, user scenario, and P turns of dialog history, each of which is a sequence of tokens. We first split the rule text x_R into several rule sentences $[x_{R,1}, x_{R,2}, ..., x_{R,M}]$ according to sentence boundary or bullet points, insert `[CLS]` tokens at the start of each sentence, and concatenate them into one sequence: [`[CLS]`, $x_{R,1}$; ... ; `[CLS]`, $x_{R,M}$; `[CLS]`, x_Q; `[CLS]`, x_S; `[CLS]`, $x_{H,1}$; ... ; `[CLS]`, $x_{H,P}$]. Then we use BERT [11], a pre-trained Vaswani-2017-AttentionIA encoder [88] to encode the sequence into a sequence of vectors with the same length. We treat each `[CLS]` representation as feature representation of the sentence that follows it. In this way, we receive both token-level representation and sentence-level representation for each sentence. We denote sentence-level representation of the rule sentences as $\mathbf{k}_1, ..., \mathbf{k}_M$ and their token-level representation as $[(\mathbf{u}_{1,1}, ..., \mathbf{u}_{1,n_1}), ..., (\mathbf{u}_{M,1}, ..., \mathbf{u}_{M,n_M})]$, where n_i is number of tokens for rule sentence i. Similarly, we denote the sentence-level representation of the initial question, user scenario, and P turns of dialog history as $\mathbf{s}_Q$, $\mathbf{s}_S$, and $\mathbf{s}_1, ..., \mathbf{s}_P$, respectively. All these vectorized representations are of d dimensions (768 for BERT-base).

6.2.2 Explicit Memory Tracking

Given the rule sentences $\mathbf{k}_1, ..., \mathbf{k}_M$ and the user provided information including the initial question $\mathbf{s}_Q$, scenario $\mathbf{s}_S$, and P turns of dialog history $\mathbf{s}_1, ..., \mathbf{s}_P$, our goal is to find implications between the rule sentences and the user provided information. Inspired by Recurrent Entity Network [26] which tracks the world state given a sequence of textual statements, we propose the **E**xplicit **M**emory **T**racker (EMT), a gated recurrent memory-augmented neural network which explicitly tracks the states of rule sentences by sequentially reading the user provided information.

As shown in Figure 6.2, EMT explicitly takes rule sentences $\mathbf{k}_1, ..., \mathbf{k}_M$ as keys, and assigns a state $\mathbf{v}_i$ to each key to save the most updated entailment information (whether this rule has been entailed from the user provided information). Each value state $\mathbf{v}_i$ is initialized with the same value of its corresponding rule sentence: $\mathbf{v}_{i,0} = \mathbf{k}_i$. Then EMT sequentially reads user provided information $\mathbf{s}_Q, \mathbf{s}_S, \mathbf{s}_1, ..., \mathbf{s}_P$. At time step t, the value state $\mathbf{v}_{i,t}$ for i-th rule sentence is updated by incorporating the user provided information $\mathbf{s}_t \in \{\mathbf{s}_Q, \mathbf{s}_S, \mathbf{s}_1, ..., \mathbf{s}_P\}$,

$$\tilde{\mathbf{v}}_{i,t} = \text{ReLU}(\mathbf{W}_k \mathbf{k}_i + \mathbf{W}_v \mathbf{v}_{i,t} + \mathbf{W}_s \mathbf{s}_t), \tag{6.1}$$

$$g_i = \sigma(\mathbf{s}_t^\top \mathbf{k}_i + \mathbf{s}_t^\top \mathbf{v}_{i,t}) \in [0, 1], \tag{6.2}$$

$$\mathbf{v}_{i,t} = \mathbf{v}_{i,t} + g_i \odot \tilde{\mathbf{v}}_{i,t} \in \mathbb{R}^d, \mathbf{v}_{i,t} = \frac{\mathbf{v}_{i,t}}{\|\mathbf{v}_{i,t}\|}, \tag{6.3}$$

where $\mathbf{W}_k, \mathbf{W}_v, \mathbf{W}_s \in \mathbb{R}^{d \times d}$, σ represents a sigmoid function, and $\odot$ is scalar product. As the user background input $\mathbf{s}_t$ may only be relevant to parts of the rule sentences, the gating function in Equation 6.2 matches $\mathbf{s}_t$ to the memory. Then EMT updates state $\mathbf{v}_{i,t}$ only in a gated manner. Finally, the normalization allows EMT to forget previous information, if necessary. After EMT sequentially reads all user provided information (the initial question, scenario, and P turns of history dialog) and finishes entailment-oriented reasoning, keys and final states of rule sentences are denoted as $(\mathbf{k}_1, \mathbf{v}_1), ..., (\mathbf{k}_M, \mathbf{v}_M)$, which will be used in the decision making module (Section 6.2.3) and question generation module (Section 6.2.4).

The key difference between our Explicit Memory Tracker and Recurrent Entity Network (REN) [26] is that each key $\mathbf{k}_i$ in our case has an explicit meaning (the corresponding rule sentence) and thus it changes according to different rule texts while in REN, the underlined meaning of keys are learned through training and they are fixed throughout all textual inputs. Moreover, the number of keys is dynamic in our case (according to the number of sentences parsed from the rule text) while that is predefined in REN.

6.2.3 Decision Making

Based on the most up-to-date key-value states of rule sentences $(\mathbf{k}_1, \mathbf{v}_1), ..., (\mathbf{k}_M, \mathbf{v}_M)$ from the EMT, the decision making module predicts a decision among `Yes`, `No`, `Irrelevant`, and `Inquire`. First, we use self-attention to compute a summary vector $\mathbf{c}$ for the overall state:

$$\alpha_i = \mathbf{w}_\alpha^\top [\mathbf{k}_i; \mathbf{v}_i] + b_\alpha \in \mathbb{R}^1, \tag{6.4}$$

$$\tilde{\alpha}_i = \text{softmax}(\alpha)_i \in [0, 1], \tag{6.5}$$

$$\mathbf{c} = \sum_i \tilde{\alpha}_i [\mathbf{k}_i; \mathbf{v}_i] \in \mathbb{R}^d, \tag{6.6}$$

where $[\mathbf{k}_i; \mathbf{v}_i]$ denotes the concatenation of the vectors $\mathbf{k}_i$ and $\mathbf{v}_i$, and α_i is the attention weight for the rule sentence $\mathbf{k}_i$

that determines the likelihood that $\mathbf{k}_i$ is entailed from the user provided information.
Then the final decision is made through a linear transformation of the summary vector $\mathbf{c}$:

$$\mathbf{z} = \mathbf{W}_z\mathbf{c} + \mathbf{b}_z \in \mathbb{R}^4, \tag{6.7}$$

where $\mathbf{z} \in \mathbb{R}^4$ contains the model's score for all four possible classes. Let l indicate the correct decision, the decision making module is trained with the following cross entropy loss:

$$\mathcal{L}_{\text{dec}} = -\log\ \text{softmax}(\mathbf{z})_l. \tag{6.8}$$

In order to explicitly track whether a condition listed in the rule has already been satisfied or not, we add a subtask to predict the entailment states for each rule sentence. The possible entailment labels are `Entailment`, `Contradiction` and `Unknown`; details of acquiring such labels are described in Section 6.3.1. With this intermediate supervision, the model can make better decisions based on the correct entailment state of each rule sentence. The entailment prediction is made through a linear transformation of the most up-to-date key-value state $[\mathbf{k}_i; \mathbf{v}_i]$ from the EMT module:

$$\mathbf{e}_i = \mathbf{W}_e[\mathbf{k}_i; \mathbf{v}_i] + \mathbf{b}_e \in \mathbb{R}^3. \tag{6.9}$$

where $\mathbf{e}_i \in \mathbb{R}^3$ contains scores of three entailment states $[\beta_{\text{entailment},i},\ \beta_{\text{contradiction},i},\ \beta_{\text{unknown},i}]$ for the i-th rule sentence. Let r indicate the correct entailment state. The entailment prediction subtask is trained with the following cross entropy loss, normalized by the number of rule sentences M:

$$\mathcal{L}_{\text{entail}} = -\frac{1}{M}\sum_{i=1}^{M}\ \log\ \text{softmax}(\mathbf{e}_i)_r. \tag{6.10}$$

6.2.4 Follow-up Question Generation

When the decision making module predicts `Inquire`, a follow-up question is required for further clarification from the user. In the same spirit of previous studies [107, 89], we decompose this problem into two stages. First, we extract a span inside the rule text which contains the underspecified user information (we name it as `underspecified span` hereafter). Second, we rephrase the extracted underspecified span into a follow-up question. We propose a coarse-to-fine approach to extract the underspecified span for the first stage, and finetune the pretrained language model UniLM [13] for the follow-up question rephrasing, as we describe below.

Coarse-to-Fine Reasoning for Underspecified Span Extraction. [107] extract the underspecified span by extracting several spans and retrieving the most likely one. The disadvantage of their approach is that extracting multiple rule spans is a challenging task, and it will propagate errors to the retrieval stage. Instead of extracting multiple spans from the rule text, we propose a coarse-to-fine reasoning approach to directly identify the underspecified span. For this, we reuse the `Unknown` scores $\beta_{\text{unknown},i}$ from the entailment prediction subtask (Eqn. 6.9), and normalize it (over the rule sentences) with a softmax to determine how likely that the i-th rule sentence contains the underspecified span:

$$\zeta_i = \text{softmax}(\beta_{\text{unknown}})_i \in [0, 1], \tag{6.11}$$

Knowing how likely a rule sentence is underspecified greatly reduces the difficulty to extract the underspecified span within it. We adopt a soft selection approach to modulate span extraction (i.e., predicting the start and end points of a span) score by rule sentence identification score ζ_i. We follow the BERTQA approach [11] to learn a start vector $\mathbf{w}_s \in \mathbb{R}^d$ and an end vector $\mathbf{w}_e \in \mathbb{R}^d$ to locate the start and end positions from the whole rule text. The probability of j-th word in i-th rule sentence $\mathbf{u}_{i,j}$ being the start/end of the span is computed as a dot product between $\mathbf{w}_s$ and $\mathbf{u}_{i,j}$, modulated by its rule sentence score ζ_i:

$$\gamma_{i,j} = \mathbf{w}_s^\top \mathbf{u}_{i,j} * \zeta_i, \quad \delta_{i,j} = \mathbf{w}_e^\top \mathbf{u}_{i,j} * \zeta_i. \tag{6.12}$$

We extract the span with the highest span score $\gamma * \delta$ under the restriction that the start and end positions must belong to the same rule sentence. Let s and e be the ground truth start and end position of the span. The underspecified span extraction loss is computed as the pointing loss

$$\mathcal{L}_{\text{span,s}} = -\mathbb{1}_{l=\text{inquire}} \log \ \text{softmax}(\gamma)_s, \tag{6.13}$$

$$\mathcal{L}_{\text{span,e}} = -\mathbb{1}_{l=\text{inquire}} \log \ \text{softmax}(\delta)_e. \tag{6.14}$$

The overall loss is the sum of the decision loss, entailment prediction loss and span extraction loss

$$\mathcal{L} = \mathcal{L}_{\text{dec}} + \lambda_1 \mathcal{L}_{\text{entail}} + \lambda_2 \mathcal{L}_{\text{span}} \tag{6.15}$$

where λ_1 and λ_2 are tunable hyperparameters.

Question Rephrasing. The underspecified span extracted in the previous stage is fed into the question rephrasing model to generate a follow-up question. We finetune the UniLM [13] to achieve this goal. UniLM is a pretrained language model which demonstrates its effectiveness in both natural language understanding and generation tasks. Specifically, it outperforms previous methods by a large margin on the SQuAD question generation task [14].

As shown in Figure 6.2, UniLM takes the concatenation of rule text and the extracted rule span as input, separated by the sentinel tokens: `[CLS]` rule-text `[SEP]` extracted-span `[SEP]`. The training target is the follow-up question we want to generate. Please refer to [13] for details on finetuning UniLM and doing inference with it.

6.3 Experiments

6.3.1 Experimental Setup

Dataset. We conduct experiments on the ShARC CMR dataset [69]. It contains 948 dialog trees, which are flattened into 32,436 examples by considering all possible nodes in the trees. Each example is a quintuple of (rule text, initial question, user scenario, dialog history, decision), where decision is either one of {`Yes`, `No`, `Irrelevant`} or a follow-up question. The train, development, and test dataset sizes are 21890, 2270, and 8276, respectively.[1]

End-to-End Evaluation. Organizers of the ShARC competition evaluate model performance as an end-to-end task. They first evaluate the micro- and macro-accuracy for the decision making task. If both the ground truth decision and the predicted decision are `Inquire`, then they evaluate the generated follow-

[1]Leaderboard: `https://sharc-data.github.io/leaderboard.html`

up question using BLEU score [59]. However, this way of evaluating follow-up questions has one issue. If two models have different `Inquire` predictions, the follow-up questions for evaluation will be different, making the comparison unfair. For example, a model could classify only one example as `Inquire` in the whole test set and generate the follow-up question correctly, achieving a 100% BLEU score. Therefore, we also propose to evaluate the follow-up question generation performance in an oracle evaluation setup as described below.

Oracle Question Generation Evaluation. In this evaluation, we ask the models to generate follow-up questions whenever the ground truth decision is `Inquire`, and compute the BLEU score for the generated questions accordingly. In this setup, there are 6804 examples for training and 562 examples for evaluation.

Data Augmentation. In the annotation process of the ShARC dataset, the *scenario* is manually constructed from a part of the dialog history, and that excerpt of the dialog is not shown as input to the model. Instead, it is treated as the *evidence* which should be entailed from the scenario. To effectively utilize this additional signal, we construct more examples by replacing the *scenario* with the *evidence*. This leads to additional 5800 training instances. We use this augmented dataset for the EMT model and its ablations in our experiments.

Labeling Underspecified Spans. To supervise the process of coarse-to-fine reasoning, we follow [107] to label the rule spans. We first trim the follow-up questions in the conversation by removing question words “do, does, did, is, was, are, have” and the question mark "?". For each trimmed question, we find the shortest span inside the rule text which has the minimum edit distance from the trimmed question, and treat it as an `underspecified span`.

Acquiring Labels for Entailment. To supervise the subtask of entailment prediction for each rule sentence, we use a heuristic to automatically label its entailment state. For each rule sentence, we first find if it contains any underspecified span for the questions in the dialog history (and evidence text), and use the corresponding `Yes/No` answers to label the rule text as `Entailment/Contradiction`. The rule text without any underspecified span is labeled as `Unknown`.

Implementation Details. We tokenize all text inputs with spaCy [28]. The EMT model and the follow-up question generation model UniLM are trained separately and pipelined together at test time. For EMT, we use the uncased BERT base model [96] for encoding. We train EMT with Adam [32] optimizer with a learning rate of 5e-5, a warm-up rate of 0.1 and a dropout rate of 0.35. The loss weights λ_1 and λ_2 in Eq. 6.15 are set to 10 and 0.6 respectively, based on the development set results. For UniLM, we fine-tuning it with a batch size of 16 and a learning rate of 2e-5, and we use a beam size of 10 for inference.

To reduce the variance of our experimental results, all experiments reported on the development set are repeated 5 times with different random seeds. We report the average results along with their standard deviations.

Table 6.1: Performance on the blind, held-out test set of ShARC end-to-end task.

Models	End-to-End Task (Leaderboard Performance)			
	Micro Acc.	Macro Acc.	BLEU1	BLEU4
Seq2Seq [69]	44.8	42.8	34.0	7.8
Pipeline [69]	61.9	68.9	54.4	34.4
BERTQA [107]	63.6	70.8	46.2	36.3
UrcaNet [89]	65.1	71.2	60.5	46.1
BiSon [37]	66.9	71.6	58.8	44.3
E^3 [107]	67.6	73.3	54.1	38.7
EMT (our single model)	**69.1**	**74.6**	**63.9**	**49.5**

Table 6.2: Class-wise decision prediction accuracy on the development set (*: reported in the paper).

Models	`Yes`	`No`	`Inquire`	`Irrelevant`
BERTQA	61.2	61.0	62.6	96.4
E^3	65.9	70.6	60.5	96.4
UrcaNet*	63.3	68.4	58.9	95.7
EMT	**70.5**	**73.2**	**70.8**	**98.6**

6.3.2 Results

End-to-End Task. The end-to-end performance on the held-out test set is shown in Table 6.1. EMT outperforms the existing state-of-the-art model E^3 on decision classification in both micro- and macro-accuracy. Although the BLEU scores are not directly comparable among different models, EMT achieves competitive BLEU1 and BLEU4 scores on the examples it makes an `Inquire` decision. The results show that EMT has strong capability in both decision making and follow-up question generation tasks. Table 6.2 presents the class-wise accuracy on the four decision types. EMT improves on the `Inquire` decision significantly. It is because EMT can explicitly track the states

Table 6.3: Performance on Oracle Question Generation Task. We show both results on the development set and 10-fold cross validation. E^3+UniLM replaces the editor of E^3 to our finetuned UniLM.

Models	Oracle Question Generation Task			
	Development Set		Cross Validation	
	BLEU1	BLEU4	BLEU1	BLEU4
E^3	52.79$\pm$2.87	37.31$\pm$2.35	51.75	35.94
E^3+UniLM	57.09$\pm$1.70	41.05$\pm$1.80	56.94	42.87
EMT	**62.32**$\pm$1.62	**47.89**$\pm$1.58	**64.48**	**52.40**

of all rule sentences; it has a macro accuracy of 80% on the entailment state prediction task.

Oracle Question Generation Task. To establish a concrete question generation evaluation, we conduct experiments on our proposed oracle question generation task. We compare our model EMT with E^3 and an extension E^3+UniLM; implementations for other methods are not publicly available. E^3+UniLM replaces the editor of E^3 with our finetuned UniLM. The results on the development set and 10-fold cross validation are shown in Table 6.3.

Firstly, E^3+UniLM performs better than E^3, validating the effectiveness of our follow-up question rephrasing module: finetuned UniLM. More importantly, EMT consistently outperforms E^3 and E^3+UniLM on both the development set and the cross validation by a large margin. Although there is no ground truth label for span extraction, we can infer from the question generation results that our coarse-to-fine reasoning approach extracts better spans than the extraction and retrieval modules of E^3. This is because E^3 propagates error from the span extraction module to the span retrieval module while our coarse-to-fine approach avoids this problem through weighting token-level span distributions with its sentence-level entailment scores.

6.3.3 Ablation Study

We conduct an ablation study on the development set for both the end-to-end evaluation task and oracle question generation evaluation task. We consider four ablations of our EMT model:

(1) EMT (w/o data aug.) trains the model on the original ShARC training set and do not use any augmented data using the evidence.

(2) EMT (w/o c2f) extracts the rule span without weighted by the entailment score ζ in Eqn. 6.12.

(3) EMT (w/o $\mathcal{L}_{\text{entail}}$) removes the entailment state prediction subtask in decision making, and thus there is no entailment score ζ for underspecified span extraction in Eqn. 6.12.

(4) EMT (w/o tracker) that removes the explicit memory tracking module. Instead, it treats the `[CLS]` token for each rule sentence as the state for decision making and span extraction.

Results of the ablations are shown in Table 6.4, and we have the following observations:

• With the help of data augmentation, EMT boosts the performance slightly on the end-to-end task, especially for the question generation task which originally has only 6804 training examples. The augmented training instances boosts the performance even though the augmentation method does not produce any new question. This implies that the size of the ShARC dataset is a bottleneck for an effective end-to-end neural models.

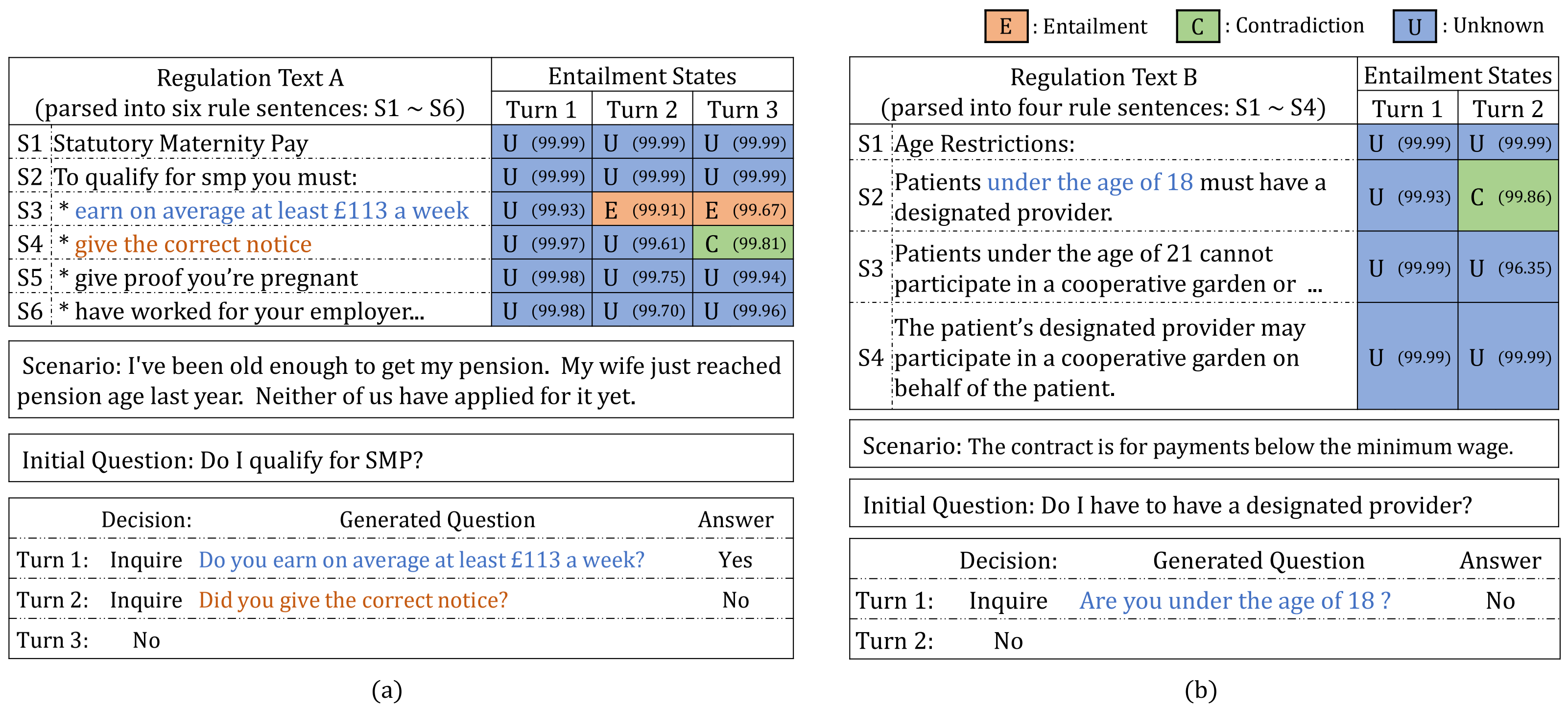
Table 6.4: Ablation Study of EMT on the development set of ShARC.

	End-to-End Task				Oracle Question Generation Task	
Models	Micro Acc.	Macro Acc.	BLEU1	BLEU4	BLEU1	BLEU4
EMT	**71.36**±0.69	**76.70**±0.54	**67.04**±1.59	**52.37**±1.92	**63.53**±1.03	**48.69**±0.80
EMT (w/o data aug.)	70.67±0.52	76.33±0.69	65.86±2.25	51.02±2.52	62.38±1.34	47.58±1.30
EMT (w/o c2f)	70.41±0.94	75.96±0.91	65.73±1.76	50.84±2.31	61.98±1.26	47.66±1.33
EMT (w/o $\mathcal{L}_{\text{entail}}$)	67.81±1.20	73.50±0.83	63.84±1.80	49.35±2.10	60.50±1.16	45.34±1.73
EMT (w/o tracker)	67.42±1.15	72.73±0.74	63.26±0.64	47.97±0.40	61.87±1.46	47.13±1.35

• Without the coarse-to-fine reasoning for span extraction, EMT (w/o c2f) drops by 1.53 on BLEU4, which implies that it is necessary for the question generation task. The reason is that, as a classification task, entailment state prediction can be trained reasonably well (80% macro accuracy) with a limited amount of data (6804 training examples). Therefore, the `Unknown` scores in the entailment state prediction can guide the span extraction via a soft modulation (Equation 6.12). On the other hand, one-step span extraction method does not utilize the entailment states of the rule sentences from EMT, meaning it does not learn to extract the underspecified part of the rule text.

• With the guidance of explicit entailment supervision, EMT outperforms EMT (w/o $\mathcal{L}_{\text{entail}}$) by a large margin. Intuitively, knowing the entailment states of the rule sentences makes the decision making process easier for complex tasks that require logical reasoning on conjunctions of conditions or disjunctions of conditions. It also helps span extraction through the coarse-to-fine approach.

• Without the explicit memory tracker described in Section 6.2.2, EMT (w/o tracker) performs poorly on the decision making task. Although there exist interactions between rule sentences and user information in BERT-encoded representations through multi-head self-attentions, it is not adequate to learn whether conditions listed in the rule text have already been satisfied or not.



🟧 E : Entailment 🟩 C : Contradiction 🟦 U : Unknown

(a)

Regulation Text A (parsed into six rule sentences: S1 ~ S6)		Entailment States		
		Turn 1	Turn 2	Turn 3
S1	Statutory Maternity Pay	U (99.99)	U (99.99)	U (99.99)
S2	To qualify for smp you must:	U (99.99)	U (99.99)	U (99.99)
S3	* earn on average at least £113 a week	U (99.93)	E (99.91)	E (99.67)
S4	* give the correct notice	U (99.97)	U (99.61)	C (99.81)
S5	* give proof you're pregnant	U (99.98)	U (99.75)	U (99.94)
S6	* have worked for your employer...	U (99.98)	U (99.70)	U (99.96)

Scenario: I've been old enough to get my pension. My wife just reached pension age last year. Neither of us have applied for it yet.

Initial Question: Do I qualify for SMP?

	Decision:	Generated Question	Answer
Turn 1:	Inquire	Do you earn on average at least £113 a week?	Yes
Turn 2:	Inquire	Did you give the correct notice?	No
Turn 3:	No		

(b)

Regulation Text B (parsed into four rule sentences: S1 ~ S4)		Entailment States	
		Turn 1	Turn 2
S1	Age Restrictions:	U (99.99)	U (99.99)
S2	Patients under the age of 18 must have a designated provider.	U (99.93)	C (99.86)
S3	Patients under the age of 21 cannot participate in a cooperative garden or ...	U (99.99)	U (96.35)
S4	The patient's designated provider may participate in a cooperative garden on behalf of the patient.	U (99.99)	U (99.99)

Scenario: The contract is for payments below the minimum wage.

Initial Question: Do I have to have a designated provider?

	Decision:	Generated Question	Answer
Turn 1:	Inquire	Are you under the age of 18 ?	No
Turn 2:	No		

Figure 6.3: Predicted decisions and generated questions by our EMT model. Extracted spans and their corresponding questions are marked in the same colors. We also visualize the transitions of predicted entailment states (**E**ntailment, **C**ontradiction, **U**nknown) over rule sentences (S1, S2, S3 ...) as the conversation flows, with associated entailment scores [$\beta_{\text{entailment}}$, $\beta_{\text{contradiction}}$, β_{unknown}].

6.3.4 Interpretability

To get better insights into the underlying entailment-oriented reasoning process of EMT, we examine the entailment states of the rule sentences as the conversation flows. Two example cases are provided in Figure 6.3. Given a rule text containing several rule sentences (S1, S2, S3, ...), we show the transition of predicted entailment states $[\beta_{\text{entailment}}, \beta_{\text{contradiction}}, \beta_{\text{unknown}}]$ over multiple turns in the dialogue.

Rules in Bullet Points. Figure 6.3 (a) shows an example in which the rule text is expressed in the conjunction of four bullet-point conditions. On the first turn, EMT reads “Scenario” and “Initial Question” and they only imply that the question from the user is relevant to the rule text. Thus the entailment states for all the rule sentences are `Unknown`, and EMT makes an `Inquire` decision, and asks a question. Once a positive answer is received from the user part for the first turn, EMT transits the entailment state for rule sentence S3 from `Unknown` to `Entailment`, but it still cannot conclude the dialogue, so it asks a second follow-up question. Then we see that the user response for the second question is negative, which makes EMT conclude a final decision `No` in the third turn.

Rules in Plain Text. Figure 6.3 (b) presents a more challenging case where the rules are in plain text. Therefore, it is not possible to put the whole sentence into a clarification question as EMT in Figure 6.3(a) does. In this case, both the decision making module and span extraction module contribute to helping the user. The span extraction module locates the correct spans inside S2, and EMT concludes a correct answer “No” after knowing the user does not fulfill the condition listed in S2.

6.3.5 Error Analysis

We analyze some errors of EMT predictions on the ShARC development set, as described below.

Decision Making Error. Out of 2270 examples in the development set, our EMT produces incorrect decisions on 608 cases. We manually analyze 104 error cases. In 40 of these cases, EMT fails to derive the correct entailment states for each rule sentence, while in 23 cases, the model predicts the correct entailment states but cannot predict correct decisions based on that. These errors suggest that explicitly modeling the logic reasoning process is a promising direction. Another challenge comes from extracting useful information from the user scenarios. In 24 cases, the model fails to make the correct decision because it could not infer necessary user information from the scenarios. Last but not least, parsing the rule text into rule sentences is also a challenge. As shown in Figure 6.3(b), the plain text usually contains complicated clauses for rule conditions, which is difficult to disentangle them into separate conditions. In 17 cases, one single rule sentence contains multiple conditions, which makes the model fail to conduct the entailment reasoning correctly.

Question Generation Error. Out of 562 question generation examples in the development set, our EMT locates the under-specified span poorly in 115 cases (span extraction F1 score

$\leq$ 0.5). We manually analyze 52 wrong question generation cases. Out of 29 cases of them, EMT fails to predict correct entailment states for rule sentences, and thus does not locate the span within the ground truth rule sentence, while in 9 cases, it finds the correct rule sentence but extracts a different span. Another challenge comes from the one-to-many problem in sequence generation. When there are multiple underspecified rule sentences, the model asks about one of these underspecified rule sentences which is different from the ground truth one. This suggests that new evaluation metrics could be proposed by taking this into consideration.

6.4 Summary

In this chapter, we have proposed a new framework for conversational machine reading (CMR) that comprises a novel explicit memory tracker (EMT) to track entailment states of the rule sentences explicitly within its memory module. The updated states are utilized for decision making and coarse-to-fine follow-up question generation in a unified manner. EMT achieved a new state-of-the-art result on the ShARC CMR challenge. EMT also gives interpretability by showing the entailment-oriented reasoning process as the conversation flows. While we conducted experiments on the ShARC dataset, we believe the proposed methodology could be extended to other kinds of CMR tasks.

□ **End of chapter.**

Chapter 7

Discourse-Aware Entailment Reasoning Network

Document interpretation and dialog understanding are the two major challenges for conversational machine reading. In this chapter, we propose DISCERN, a discourse-aware entailment reasoning network to strengthen the connection and enhance the understanding for both document and dialog. We first give an introduction in Section 7.1. Then we describe our proposed Discern DISCERN in Section 7.2. Section 7.3 describe our experimental setting and results. Finally, we give a summary of this chapter in Section 7.4.

7.1 Introduction

Conversational Machine Reading (CMR) is challenging because the rule text may not contain the literal answer, but provide a procedure to derive it through interactions [69]. In this case, the machine needs to read the rule text, interpret the user scenario, clarify the unknown user's background by asking questions, and derive the final answer. Taking Figure 7.1 as an example, to

Rule Text: 7(a) loans are the most basic and most used type loan of the Small Business Administration's (SBA) business loan programs. It's name comes from section 7(a) of the Small Business Act, which authorizes the agency to provide business loans to **American small businesses**. The loan program is designed to assist **for-profit businesses** that are **not able to get other financing from other resources**.

User Scenario: I am a 34 year old man from the United States who owns their own business. We are an American small business.
User Question: Is the 7(a) loan program for me?
Follow-up $\mathbf{Q}_1$: Are you a for-profit business?
Follow-up $\mathbf{A}_1$: Yes.
Follow-up $\mathbf{Q}_2$: Are you able to get financing from other resources?
Follow-up $\mathbf{A}_2$: No.
Final Answer: Yes. (You can apply the loan.)

Figure 7.1: An example dialog from the ShARC [69] dataset. The machine answers the user question by reading the rule text, interpreting the user scenario, and keeping asking follow-up questions to clarify the user's background until it concludes a final answer. Requirements in the rule text are bold.

answer the user whether he is suitable for the loan program, the machine needs to interpret the rule text to know what are the requirements, understand he meets "American small business" from the user scenario, ask follow-up clarification questions about "for-profit business" and "not get financing from other resources", and finally it concludes the answer "Yes" to the user's initial question.

Existing approaches [107, 89, 20] decompose this problem into two sub-tasks. Given the rule text, user question, user scenario, and dialog history (if any), the first sub-task is to make a decision among "Yes", "No", "Inquire" and "Irrelevant". The "Yes/No" directly answers the user question and "Irrelevant" means the user question is unanswerable by the rule text. If the user-provided information (user scenario, previous dialogs) are not enough to determine his fulfillment or eligibility, an "Inquire" decision is made and the second sub-task is activated. The second sub-task is to capture the underspecified condition from the rule text and generate a follow-up question to clarify it. [107] adopt BERT [11] to reason out the decision, and propose an entailment-driven extracting and editing framework to extract a span from the rule text and edit it into the follow-up question. The current state-of-the-art model EMT [20] uses a Recurrent Entity Network [26] with explicit memory to track the fulfillment of rules at each dialog turn for decision making and question generation.

In this problem, document interpretation requires identification of conditions and determination of logical structures because rules can appear in the format of bullet points, in-line conditions, conjunctions, disjunctions, etc. Hence, correctly interpreting rules is the first step towards decision making. Another challenge is dialog understanding. The model needs to evaluate the user's fulfillment over the conditions, and jointly consider the fulfillment states and the logical structure of rules for decision making. For example, disjunctions and conjunctions of conditions have completely different requirements over the user's fulfillment states. However, existing methods have not considered condition-level understanding and reasoning.

In this chapter, we propose DISCERN: **Disc**ourse-Aware **E**ntailment **R**easoning **N**etwork . To better understand the logical structure of a rule text and to extract conditions from it, we first segment the rule text into clause-like elementary discourse units (EDUs) using a pre-trained discourse segmentation model [39]. Each EDU is treated as a condition of the rule text, and our model estimates its entailment confidence scores over three states: Entailment, Contradiction or Neutral by reading the user scenario description and existing dialog. Then we map the scores to an entailment vector for each condition, and reason out the decision based on the entailment vectors and the logical structure of rules. Compared to previous methods that do little entailment reasoning [107] or use it as multi-task learning [20], Discern is the first method to explicitly build the dependency between entailment states and decisions at each dialog turn.

Discern achieves new state-of-the-art results on the blind, held out test set of ShARC [69]. In particular, Discern outperforms the previous best model EMT [20] by 3.8% in micro-averaged decision accuracy and 3.5% in macro-averaged decision accuracy. Specifically, Discern performs well on simple in-line conditions and conjunctions of rules while still needing improvements on understanding disjunctions. Finally, we conduct comprehensive analyses to unveil the limitation of Discern and current challenges for the ShARC benchmark. We find one of the biggest bottlenecks is the user scenario interpretation, in which various types of reasoning are required.

7.2 Discern Model

Discern answers the user question through a three-step process shown in Figure 7.2:

1. First, Discern segments the rule text into individual conditions using discourse segmentation.
2. Taking the user-provided information including the user question, user scenario and dialog history as inputs, Discern

predicts the entailment state and maps it to an entailment vector for each segmented condition. Then it reasons out the decision by considering the logical structure of the rule text and the fulfillment of each condition.

3. Finally, if the decision is "Inquire", DISCERN generates a follow-up question to clarify the underspecified condition in the rule text.

7.2.1 Rule Segmentation

The goal of rule segmentation is to understand the logical structure of the rule text and parse it into individual conditions for the ease of entailment reasoning. Ideally, each segmented unit should contain at most *one* condition. Otherwise, it will be ambiguous to determine the entailment state for that unit. Determining conditions is easy when they appear as bullet points, but in most cases (65% samples in the ShARC dataset), one rule sentence may contain several in-line conditions as exemplified in Figure 7.2. To extract these in-line conditions, we find discourse segmentation in discourse parsing to be useful. In the Rhetorical Structure Theory or RST [52] of discourse parsing, texts are first split into a sequence of clause-like units called elementary discourse units (EDUs). We utilize an off-the-shelf discourse segmenter [39] to break the rule text into a sequence of EDUs. The segmenter uses a pointer network and achieves 92.2% F-score with Glove vectors and 95.55% F-score with ELMo embeddings on the standard RST benchmark testset, which is close to human agreement of 98.3% F-score [30, 46]. As exemplified in Figure 7.2 Step ①, the rule sentence

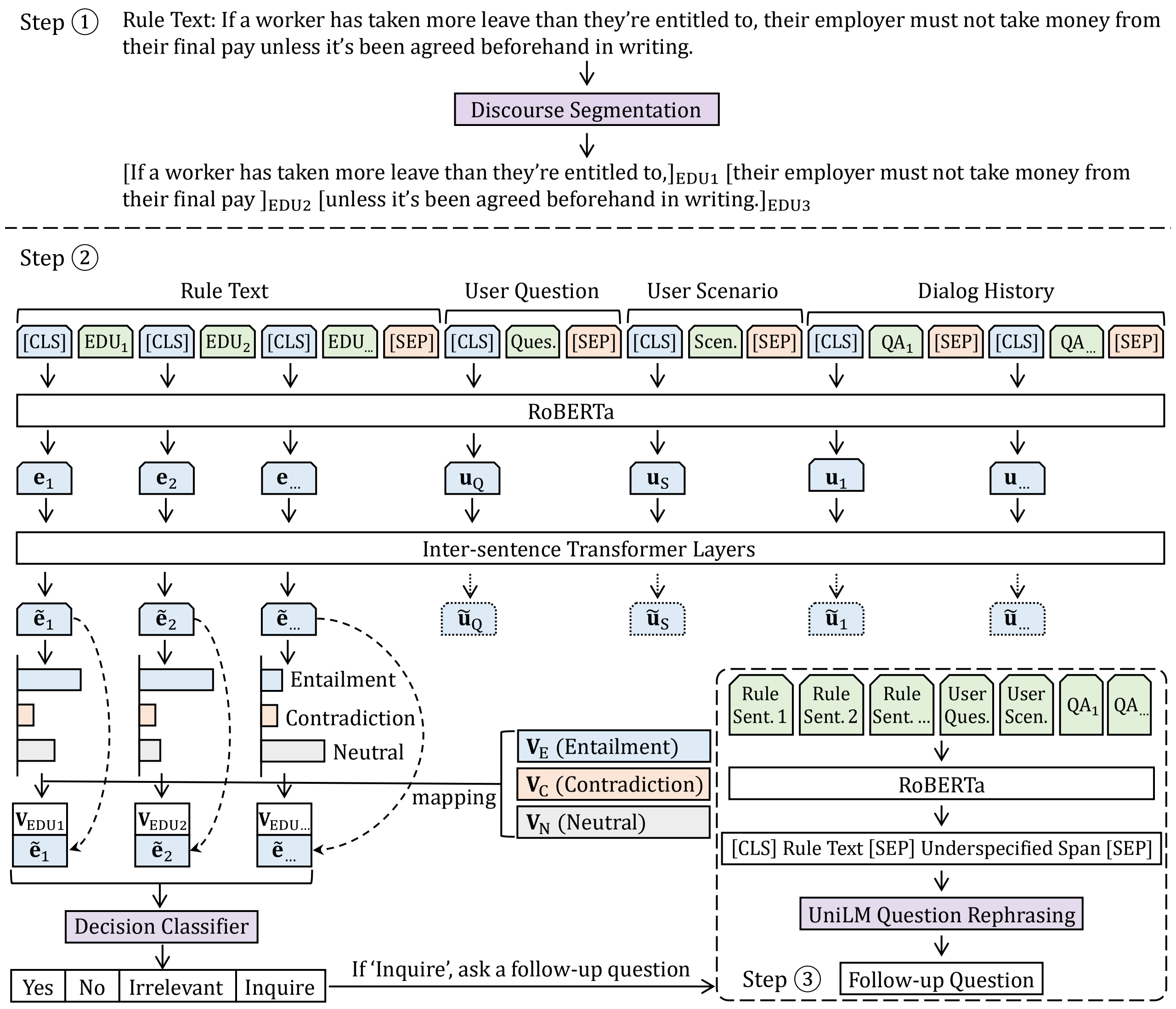

Step ①
Rule Text: If a worker has taken more leave than they're entitled to, their employer must not take money from their final pay unless it's been agreed beforehand in writing.
Discourse Segmentation
[If a worker has taken more leave than they're entitled to,]$_{EDU1}$ [their employer must not take money from their final pay]$_{EDU2}$ [unless it's been agreed beforehand in writing.]$_{EDU3}$
Step ②
Rule Text
User Question
User Scenario
Dialog History
[CLS]
EDU$_1$
EDU$_2$
EDU$_{...}$
[SEP]
Ques.
Scen.
QA$_1$
QA$_{...}$
RoBERTa
$\mathbf{e}_1$
$\mathbf{e}_2$
$\mathbf{e}_{...}$
$\mathbf{u}_Q$
$\mathbf{u}_S$
$\mathbf{u}_1$
$\mathbf{u}_{...}$
Inter-sentence Transformer Layers
$\tilde{\mathbf{e}}_1$
$\tilde{\mathbf{e}}_2$
$\tilde{\mathbf{e}}_{...}$
$\tilde{\mathbf{u}}_Q$
$\tilde{\mathbf{u}}_S$
$\tilde{\mathbf{u}}_1$
$\tilde{\mathbf{u}}_{...}$
Entailment
Contradiction
Neutral
$\mathbf{V}_E$ (Entailment)
$\mathbf{V}_C$ (Contradiction)
$\mathbf{V}_N$ (Neutral)
mapping
$\mathbf{V}_{EDU1}$
$\mathbf{V}_{EDU2}$
$\mathbf{V}_{EDU...}$
Decision Classifier
Yes
No
Irrelevant
Inquire
If 'Inquire', ask a follow-up question
Rule Sent. 1
Rule Sent. 2
Rule Sent. ...
User Ques.
User Scen.
RoBERTa
[CLS] Rule Text [SEP] Underspecified Span [SEP]
UniLM Question Rephrasing
Step ③
Follow-up Question

Figure 7.2: The overall diagram of our proposed DISCERN. DISCERN first segments the rule text into several elementary discourse units (EDUs) as conditions (Section 7.2.1). Then, taking the segmented conditions, user question, user scenario, and dialog history as inputs, DISCERN reasons out the decision among "Yes", "No", "Irrelevant" and "Inquire" (Section 7.2.2). If the decision is "Inquire", the question generation model asks a follow-up question (Section 7.2.3). *(Best viewed in color)*

is broken into three EDUs, in which two conditions ("If a worker has taken more leave than they're entitled to", "unless it's been agreed beforehand in writing") and the outcome ("their employer must not take money from their final pay") are split out precisely. For rule texts which contain bullet points, we directly treat these bullet points as conditions.

7.2.2 Decision Making via Entailment Reasoning

Encoding. As shown in Figure 7.2 Step ②, inputs to DISCERN include the segmented conditions (EDUs) in the rule text, user question, user scenario, and follow-up question-answer pairs in dialog history, each of which is a sequence of tokens. In order to get the sentence-level representations for all individual sequences, we insert an external `[CLS]` symbol at the start of each sequence, and add a `[SEP]` symbol at the end of every type of inputs. Then, DISCERN concatenates all sequences together, and uses RoBERTa [49] to encode the concatenated sequence. The encoded `[CLS]` token represents the sequence that follows it. In this way, we extract sentence-level representations of conditions (EDUs) as $\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_N$, and also the representations of the user question $\mathbf{u}_Q$, user scenario $\mathbf{u}_S$, and M turns of dialog history $\mathbf{u}_1, ..., \mathbf{u}_M$. All these vectorized representations are of d dimensions (768 for RoBERTa-base).

Entailment Prediction. In order to reason out the correct decision for the user question, it is necessary to figure out the fulfillment of conditions in the rule text. We propose to formulate the fulfillment prediction of conditions into a multi-sentence entailment task. Given a sequence of conditions (premises) and a sequence of user-provided information (hypotheses), a system should output ENTAILMENT, CONTRADICTION or NEUTRAL for each condition listed in the rule text. In this context, NEUTRAL indicates that the condition has not been mentioned from the user information.

We utilize an inter-sentence transformer encoder [88] to predict the entailment states for all conditions simultaneously. Taking all sentence-level representations $[\mathbf{e}_1; \mathbf{e}_2; ...; \mathbf{e}_N; \mathbf{u}_Q; \mathbf{u}_S; \mathbf{u}_1; ...; \mathbf{u}_M]$ as inputs, the L-layer transformer encoder makes each condition attend to all the user-provided information to predict whether the condition is entailed or not. We also allow all conditions can attend to each other to understand the logical structure of the rule text.

Let the transformer encoder output of the i-th condition as $\tilde{\mathbf{e}}_i$, we use a linear transformation to predict its entailment state:

$$\mathbf{c}_i = \mathbf{W}_c \tilde{\mathbf{e}}_i + \mathbf{b}_c \in \mathbb{R}^3, \tag{7.1}$$

where $\mathbf{c}_i = [c_{\mathrm{E},i}, c_{\mathrm{C},i}, c_{\mathrm{N},i}] \in \mathbb{R}^3$ contains confidence scores of three entailment states ENTAILMENT, CONTRADICTION, NEUTRAL for the i-th condition in the rule text.

Since there are no ground truth entailment labels for individual conditions, we adopt a heuristic approach similar to [20] to get the noisy supervision signals. Given the rule text, we first collect all associated follow-up questions in the dataset. Each follow-up question is matched to a segmented condition (EDU) in the rule text which has the minimum edit distance. For conditions in the rule text which are mentioned by follow-up questions in the dialogue history, we label the entailment state of a condition as `Entailment` if the answer for its mentioned follow-up question is `Yes`, and label the state of this condition as `Contradiction` if the answer is `No`. The remaining conditions not covered by any follow-up question are labeled as `Neutral`. Let r indicate the correct entailment state. The entailment prediction is weakly supervised by the following cross entropy loss, normalized by total number of K conditions in a batch:

$$\mathcal{L}_{\text{entail}} = -\frac{1}{K}\sum_{i=1}^{K} \log \ \text{softmax}(\mathbf{c}_i)_r. \tag{7.2}$$

Decision Making. After knowing the entailment state for each condition in the rule text, the remaining challenge for decision making is to perform logical reasoning over different rule types such as disjunction, conjunction, and conjunction of disjunctions. To achieve this, we first design three d-dimension entailment vectors $\mathbf{V}_{\text{E}}$ (Entailment), $\mathbf{V}_{\text{C}}$ (Contradiction), $\mathbf{V}_{\text{N}}$ (Neutral), and map the predicted entailment confidence scores of each condition to its vectorized entailment representation:

$$\mathbf{V}_{\text{EDU},i} = \sum_{k\in[\text{E,C,N}]} c_{k,i}\mathbf{V}_{\text{k}} \in \mathbb{R}^d, \tag{7.3}$$

These entailment vectors are randomly initialized and then learned during training. Finally, DISCERN jointly considers the logical structure of rules $\tilde{\mathbf{e}}_i$ and the entailment representations $\mathbf{V}_{\text{EDU},i}$ of conditions to make a decision:

$$\alpha_i = \mathbf{w}_\alpha^\top[\mathbf{V}_{\text{EDU},i};\tilde{\mathbf{e}}_i] + b_\alpha \in \mathbb{R}^1, \tag{7.4}$$

$$\tilde{\alpha}_i = \text{softmax}(\alpha)_i \in [0,1], \tag{7.5}$$

$$\mathbf{g} = \sum_i \tilde{\alpha}_i[\mathbf{V}_{\text{EDU},i};\tilde{\mathbf{e}}_i] \in \mathbb{R}^{2d}, \tag{7.6}$$

$$\mathbf{z} = \mathbf{W}_z\mathbf{g} + \mathbf{b}_z \in \mathbb{R}^4, \tag{7.7}$$

where $[\mathbf{V}_{\text{EDU},i}; \tilde{\mathbf{e}}_i]$ denotes the vector concatenation, α_i is the attention weight for the i-th condition that determines whether the i-th condition should be taken into consideration for the final decision. $\mathbf{z} \in \mathbb{R}^4$ contains the predicted scores for all four possible decisions "Yes", "No", "Inquire" and "Irrelevant". Let l indicate the correct decision, $\mathbf{z}$ is supervised by the following cross entropy loss:

$$\mathcal{L}_{\text{dec}} = -\log \ \text{softmax}(\mathbf{z})_l. \tag{7.8}$$

The overall loss for the Step ② decision making is the weighted-sum of decision loss and entailment prediction loss:

$$\mathcal{L} = \mathcal{L}_{\text{dec}} + \lambda \mathcal{L}_{\text{entail}}. \tag{7.9}$$

7.2.3 Follow-up Question Generation

If the predicted decision is "Inquire", the follow-up question generation model is activated, as shown in Step ③ of Figure 7.2. It extracts an *underspecified* span from the rule text which is uncovered from the user's feedback, and rephrases it into a well-formed question. Existing approaches put huge efforts in extracting the underspecified span, such as entailment-driven extracting and ranking [107] or coarse-to-fine reasoning [20]. However, we find that such sophisticated modelings may not be necessary, and we propose a simple but effective approach here.

We split the rule text into sentences and concatenate the rule sentences and user-provided information into a sequence. Then we use RoBERTa to encode them into vectors grounded to tokens, as here we want to predict the position of a span within the rule text. Let $[\mathbf{t}_{1,1}, ..., \mathbf{t}_{1,s_1}; \mathbf{t}_{2,1}, ..., \mathbf{t}_{2,s_2}; ...; \mathbf{t}_{N,1}, ...,$

$\mathbf{t}_{N,s_N}]$ be the encoded vectors for tokens from N rule sentences, we follow the BERTQA approach [11] to learn a start vector $\mathbf{w}_s \in \mathbb{R}^d$ and an end vector $\mathbf{w}_e \in \mathbb{R}^d$ to locate the start and end positions, under the restriction that the start and end positions must belong to the same rule sentence:

$$\text{Span} = \arg\max_{i,j,k}(\mathbf{w}_s^\top \mathbf{t}_{k,i} + \mathbf{w}_e^\top \mathbf{t}_{k,j}), \tag{7.10}$$

where i, j denote the start and end positions of the selected span, and k is the sentence which the span belongs to. The training objective is the sum of the log-likelihoods of the correct start and end positions. To supervise the span extraction process, the noisy supervision of spans are generated by selecting the span which has the minimum edit distance with the to-be-asked question. Lastly, following [20], we concatenate the rule text and span as the input sequence, and finetune UniLM [13], a pre-trained language model to rephrase it into a question.

7.3 Experiments

7.3.1 Experimental Setup

Dataset. ShARC [69] dataset is the current benchmark to test entailment reasoning in conversational machine reading [1]. The dataset contains 948 rule texts clawed from 10 government websites, in which 65% of them are plain text with in-line conditions while the rest 35% contain bullet-point conditions. Each rule text is associated with a dialog tree (follow-up QAs) that considers all possible fulfillment combinations of conditions. In the data annotation stage, parts of the dialogs are

[1]Leaderboard: `https://sharc-data.github.io/leaderboard.html`

paraphrased into the user scenario. These parts of dialogs are marked as `evidence` which should be extracted (entailed) from the user scenario, and are not provided as inputs for evaluation. The inputs to the system are the rule text, user question, user scenario, and dialog history (if any). The output is the answer among Yes, No, Irrelevant, or a follow-up question. The train, development, and test dataset sizes are 21890, 2270, and 8276, respectively.

Evaluation Metrics. The decision making sub-task uses macro- and micro- accuracy of four classes "Yes", "No", "Irrelevant", "Inquire" as metrics. For the question generation sub-task, we evaluate models under both the official end-to-end setting [69] and the recently proposed oracle setting [20]. In the official setting, the BLEU score [59] is calculated only when both the ground truth decision and the predicted decision are "Inquire", which makes the score dependent on the model's "Inquire" predictions. For the oracle question generation setting, models are asked to generate a question when the ground truth decision is "Inquire".

Implementation Details. For the decision making sub-task, we finetune RoBERTa-base model [96] with Adam optimizer for 5 epochs with a learning rate of 5e-5, a warm-up rate of 0.1, a batch size of 16, and a dropout rate of 0.35. The number of inter-sentence transformer layers L and the loss weight λ for entailment prediction are hyperparameters. We try 1,2,3 for L and 1.0, 2.0, 3.0, 4.0, 5.0 for λ, and find the best combination is $L = 2, \lambda = 3.0$, based on the development set results. For the question generation sub-task, we train a RoBERTa-base model

Table 7.1: Performance on the blind, held-out test set of ShARC end-to-end task.

Models	End-to-End Task (Leaderboard Performance)			
	Micro Acc.	Macro Acc.	BLEU1	BLEU4
Seq2Seq [69]	44.8	42.8	34.0	7.8
Pipeline [69]	61.9	68.9	54.4	34.4
BERTQA [107]	63.6	70.8	46.2	36.3
UrcaNet [89]	65.1	71.2	60.5	46.1
BiSon [37]	66.9	71.6	58.8	44.3
E^3 [107]	67.6	73.3	54.1	38.7
EMT [20]	69.4	74.8	60.9	46.0
EMT+entailment [20]	69.1	74.6	63.9	**49.5**
DISCERN (our single model)	**73.2**	**78.3**	**64.0**	49.1

to extract spans under the same training scheme above, and finetune UniLM [13] 20 epochs for question rephrasing with a batch size of 16, a learning rate of 2e-5, and a beam size 10 for decoding in the inference stage. We repeat 5 times with different random seeds for all experiments on the development set and report the average results along with their standard deviations. It takes two hours for training on a 4-core server with an Nvidia GeForce GTX Titan X GPU.

7.3.2 Results

Decision Making Sub-task. The decision making results in macro- and micro- accuracy on the blind, held out test set of ShARC are shown in Table 7.1. DISCERN outperforms the previous best model EMT [20] by 3.8% in micro-averaged accuracy and 3.5% in macro-averaged accuracy. We further analyze the class-wise decision prediction accuracy on the development set of ShARC in Table 7.2, and find that DISCERN have far better predictions than all existing approaches whenever a decision on the user's fulfillment is needed ("Yes", "No", "Inquire"). It is because the

Table 7.2: Class-wise decision prediction accuracy among "Yes", "No", "Inquire" and "Irrelevant" on the development set of ShARC.

Models	Yes	No	Inq.	Irr.
BERTQA	61.2	61.0	62.6	96.4
E^3	65.9	70.6	60.5	96.4
UrcaNet	63.3	68.4	58.9	95.7
EMT	70.5	73.2	70.8	98.6
DISCERN	**71.9**	**75.8**	**73.3**	**99.3**

predicted decisions from DISCERN are made upon the predicted entailment states while previous approaches do not build the connection between them.

Question Generation Sub-task. DISCERN outperforms existing methods under both the official end-to-end setting (Table 7.1) and the recently proposed oracle setting (Table 7.3). Because the comparison among models is only fair under the oracle question generation setting [20], we compare DISCERN with E^3 [107], E^3+UniLM [20], EMT [20], and our ablation DISCERN (BERT) in Table 7.3. Interestingly, we find that, in this oracle setting, our proposed simple approach is even better than previous sophisticated models such as E^3 and EMT which jointly learn question generation and decision making via multi-task learning. From our results and investigations, we believe the decision making sub-task and the follow-up question generation sub-task do not share too many commonalities so the results are not improved for each task in their multi-task training. On the other hand, our question generation model is easy to optimize because this model is separately trained from the decision making one, which means there is no need to balance the performance between these two sub-tasks. Besides, RoBERTa

Table 7.3: Oracle question generation performance on the development set of ShARC.

Models	BLEU1	BLEU4
E^3	52.79±2.87	37.31±2.35
E^3+UniLM	57.09±1.70	41.05±1.80
EMT	62.32±1.62	47.89±1.58
DISCERN (BERT)	64.13±0.43	50.73±0.72
DISCERN	**64.23**±0.84	**50.85**±0.89

backbone performs comparably with its BERT counterpart.

In our detailed analyses, we find DISCERN can locate the next questionable sentence with 77.2% accuracy, which means DISCERN utilizes the user scenario and dialog history well to locate the next underspecified condition. We try to add entailment prediction supervision to help DISCERN to locate the unfulfilled condition but it does not help. We also try to simplify our approach by directly finetuning UniLM to learn the mapping between concatenated input sequences and the follow-up clarification questions. However, the poor result (around 40 for BLEU1) suggests this direction still remains further investigations.

7.3.3 Ablation Study

Table 7.4 shows an ablation study of DISCERN for the decision making sub-task on the development set of ShARC, and we have the following observations:

RoBERTa *vs.* BERT. DISCERN (BERT) replaces the RoBERTa backbone with BERT while other modules remain the same. The better performance of RoBERTa backbone matches findings

Table 7.4: Ablation study of DISCERN for decision making on the development set of ShARC.

Models	Micro Acc.	Macro Acc.
DISCERN	74.97±0.27	79.55±0.35
DISCERN (BERT)	73.07±0.21	77.77±0.24
DISCERN (w/o EDU)	73.34±0.22	78.25±0.57
DISCERN (w/o Trans)	74.25±0.36	78.78±0.57
DISCERN (w/o $\tilde{\mathbf{e}}$)	73.55±0.26	78.19±0.30
DISCERN (w/o $\mathbf{V}_{\text{EDU}}$)	72.95±0.23	77.53±0.19

from [83], which indicate that RoBERTa can capture negations and handle conjunctions of facts better than BERT.

Discourse Segmentation *vs.* Sentence Splitting. DISCERN (w/o EDU) replaces the discourse segmentation based rule parsing with simple sentence splitting, and we observe there is a 1.63% drop on the micro-accuracy. This is intuitive because we observe 65% of the rule texts in the training set contains in-line conditions. To better understand the effect of discourse segmentation, we also evaluate DISCERN and DISCERN (w/o EDU) on just that portion of examples that contains multiple EDUs. The micro-accuracy of decision making is 75.75 for DISCERN while it is 70.98 for DISCERN (w/o EDU). The significant gap shows that discourse segmentation is extremely helpful.

Are Inter-sentence Transformer Layers Necessary? We investigate the necessity of inter-sentence transformer layers because RoBERTa-base already has 12 transformer layers, in which the sentence-level `[CLS]` representations can also interact with each other via multi-head self-attention. Therefore, we remove the inter-sentence transformer layers and use the RoBERTa encoded

Table 7.5: Decision prediction accuracy categorized by logical types of rules on the ShARC development set.

Logical Type	# samples	Micro Acc.	Macro Acc.
Simple	569	82.78$\pm$1.48	86.91$\pm$1.31
Disjunction	726	69.97$\pm$1.85	75.89$\pm$1.38
Conjunction	698	74.47$\pm$2.41	79.78$\pm$1.74
Other	277	73.29$\pm$2.53	77.17$\pm$1.54

`[CLS]` representations for entailment prediction and decision making. The results show that removing the inter-sentence transformer layers (Discern w/o Trans) hurts the performance, which suggests that the inter-sentence self-attention is essential.

Both Condition Representations and Entailment Vectors Facilitate Decisions. We remove either the condition representations $\tilde{\mathbf{e}}_i$ or the entailment vectors $\mathbf{V}_{\text{EDU}}$ in Eqn.7.4 & 7.6 for decision predictions. The results show that both sides of the information are useful for making decisions. Presumably, the condition representations account for the logical forms of rule texts and entailment vectors contain the fulfillment states for these conditions.

7.3.4 Analysis of Logical Structure of Rules

To see how Discern understands the logical structure of rules, we evaluate the decision making accuracy according to the logical types of rule texts. Here we define four logical types: "Simple", "Conjunction", "Disjunction", "Other", which are inferred from the associated dialog trees. "Simple" means there is only one requirement in the rule text while "Other" denotes the rule text have complex logical structures, for example, a conjunction of disjunctions or a disjunction of conjunctions. Table 7.5 shows decision prediction results categorized by different logical structures of rules. Discern achieves the best performance on the "Simple" logical type which only needs to determine the single condition is satisfied or not. On the other hand, Discern does not perform well on rules in the format of disjunctions. We conduct further analysis on this category and find that the error comes from user scenario interpretation: the user has already provided his fulfillment in the user scenario but Discern fails to extract it. Detailed analyses are further conducted in the following section.

7.3.5 How Far Has the Problem Been Solved?

In order to figure out the limitations of Discern, and the current challenges of ShARC CMR, we disentangle the challenges of scenario interpretation and dialog understanding in ShARC by selecting different subsets, and evaluate decision making and entailment prediction accuracy on them.

Baseline. Because the classification for unanswerable questions ("irrelevant" class) is nearly solved (99.3% in Table 7.2), we create the baseline subset by removing all unanswerable examples from the development set. Results for this baseline are shown in **ShARC (Answerable)** of Table 7.6.

Dialog History Subset. We first want to see how Discern understands dialog histories (follow-up QAs) without the influence of user scenarios. Hence, we create a subset of ShARC (Answerable) in which all samples have an empty user scenario. The

Table 7.6: Decision making and entailment prediction results over different subsets of the ShARC development set.

Dataset	Decision Making		Entailment Prediction	
	Micro Acc.	Macro Acc.	Micro Acc.	Macro Acc.
ShARC (Answerable)	73.55±0.33	73.46±0.27	86.41±0.39	81.13±0.39
Dialog History Subset	79.29±1.62	76.37±1.95	92.41±0.38	90.12±0.68
Scenario Subset	63.50±1.58	60.18±1.72	82.76±0.46	59.40±1.04
ShARC (Evidence)	84.93±0.29	84.37±0.24	91.46±0.68	89.90±1.40

performance over 224 such samples is shown in "Dialog History Subset" of Table 7.6. Surprisingly, the results on this portion of samples are much better than the overall results, especially for the entailment prediction (92.41% micro-accuracy).

Scenario Subset. With the curiosity to see what is the bottleneck of our model, we test the model ability on scenario interpretation. Similarly, we create a "Scenario Subset" from ShARC (Answerable) in which all samples have an empty dialog history. Results in Table 7.6 ("Scenario Subset") show that interpreting scenarios to extract the entailment information within is exactly the current bottleneck of Discern. We analyze 100 error cases on this subset and find that various types of reasoning are required for scenario interpretation, including numerical reasoning (15%), temporal reasoning (12%), and implication over common sense and external knowledge (46%). Besides, Discern still fails to extract user's fulfillment when the scenarios paraphrase the rule texts (27%). Examples for each type of error are shown in Figure 7.3. Among three classes of entailment states, we find that Discern fails to predict Entailment or Contradiction precisely – it predicts Neutral in most cases for scenario interpretation, resulting in

Error Type	%	Example
Numerical Reasoning	15	**Relevant Rule**: Each attachment must be less than 10MB. **Scenario**: The attachment right now isn't less than 10MB, but I think I can compress so it becomes less than 10MB. **Question**: Can I upload the attachment? **Entailment State** Gold: `Entailment`; Predict: `Contradiction`
Temporal Reasoning	12	**Relevant Rule**: The Additional State Pension is an extra amount of money you could get on top of your basic State Pension if you're: ... * a woman born before 6 April 1953 **Scenario**: I live with my husband. We both have worked all our lives. Both of us were born in 1950. **Question**: Can I get Additional State Pension? **Entailment State** Gold: `Entailment`; Predict: `Contradiction`
Commonsense Reasoning	46	**Relevant Rule**: Homeowners may apply for up to $200,000 to repair or replace their primary residence to its pre-disaster condition. **Scenario**: My home was flooded **Question**: Is this loan suitable for me? **Entailment State** Gold: `Entailment`; Predict: `Unknown`
Paraphrase Reasoning	27	**Relevant Rule**: The Montgomery GI Bill (MGIB) is an educational assistance program enacted by Congress to attract high quality men and women into the Armed Forces. **Scenario**: I applied and found out I can get a loan. My dad wants me to join the army, but I don't. I'd rather go to school. **Question**: Does this program meet my needs? **Entailment State** Gold: `Contradiction`; Predict: `Unknown`

Figure 7.3: Types of scenario interpretation errors in the development data based on 100 samples.

high micro-accuracy in entailment prediction but the macro-accuracy is poor. The decision accuracy is subsequently hurt by the entailment results.

ShARC (Evidence). Based on the above observation, we replace the user scenario in the ShARC (Answerable) by its `evidence` and re-evaluate the overall performance on these answerable questions. As described in Section 7.3.1 Dataset, the `evidence` is the part of dialogs that should be entailed from the user scenario. Table 7.6 shows that the model improves 11.38% in decision making micro-accuracy if no scenario interpretation is required, which validates our above observation.

7.4 Summary

In this chapter, we present DISCERN, a system that does discourse-aware entailment reasoning for conversational machine reading. DISCERN explicitly builds the connection between entailment states of conditions and the final decisions. Results on the ShARC benchmark shows that DISCERN outperforms existing methods by a large margin. We also conduct comprehensive analyses to unveil the limitations of DISCERN and challenges for ShARC.

□ **End of chapter.**

Chapter 8

Conclusion and Future Work

In this chapter, we first summarize the contributions of this thesis, and then we present potential future research directions.

8.1 Conclusion

The capacity to use complex language to communicate and maintain our social world has been a trademark of humanity. Among different goals of language usage, questions and answers play significant roles in our day-to-day communications. We *ask* questions to explore unknown information from our side, and we *answer* questions to fill the information gap of others. In this thesis, we introduce our efforts to make machines ask and answer questions with human. We do not limit our research scope to a specific research topic such as question answering, question generation, or dialogue. Instead, our research comes from two important communication needs of human: Knowledge Assessment and Information Acquisition.

In the first part of this thesis, we investigate how machines can test knowledge of human. Our research focuses on three aspects listed in Chapter 3, Chapter 4, and Chapter 5. Specifically, our contributions can make the generated question difficulty controllable, make the generated options in multiple-choice questions more distracting, and make the question asking process more conversational.

In the second part of this thesis, we stand from the information gathering perspective for question answering research. Our research contributions are described in Chapter 6 and Chapter 7. Specifically, we focus on conversational machine reading, in which machines need to answer some high-level questions by asking some clarification follow-up questions first. Therefore, both asking and answering questions are important to build satisfying conversation machine reading systems. Our contributions in the second part are two folds: 1) we propose an explicit memory tracker to track the dialogue states in conversations. 2) we propose to leverage discourse parsing to understand the rule text and entailment reasoning over the dialogue states for conversational machine reading.

8.2 Future Work

Question answering is a promising research area and has received rapid progress in the era of deep learning. Despite the contributions proposed in this chapter, there are still plenty of exciting research directions that are valuable to explore in the future.

Open-Retrieval Conversational Machine Reading. In Chapter 6 and Chapter 7, the machine reading system answers the user question according to the given rule text. However, it neglects

Retrieved Rule Text 1: SBA provides loans **to businesses - not individuals** - so the requirements of eligibility are based on aspects of the business, not the owners. All businesses that are considered for financing under SBA's 7(a) loan program must: **meet SBA size standards**, be **for-profit**, **not already have the internal resources (business or personal) to provide the financing**, and be able to demonstrate repayment.
Retrieved Rule Text 2: You'll need a statement of National Insurance you've paid in the UK to get these benefits - unless you're claiming Winter Fuel Payments.
Retrieved Rule Text 3: 7(a) loans are the most basic and most used type loan of the Small Business Administration's (SBA) business loan programs. It's name comes from section 7(a) of the Small Business Act, which authorizes the agency to provide business loans to **American small businesses**. The loan program is designed to assist **for-profit businesses** that are **not able to get other financing from other resources**.

User Scenario: I am a 34 year old man from the United States who owns their own business. We are an American small business.
User Question: Is the 7(a) loan program for me?
Follow-up $\mathbf{Q}_1$: Are you a for-profit business?
Follow-up $\mathbf{A}_1$: Yes.
Follow-up $\mathbf{Q}_2$: Are you able to get financing from other resources?
Follow-up $\mathbf{A}_2$: No.
Final Answer: Yes. (You can apply the loan.)

Figure 8.1: Open retrieval conversational machine reading task. The machine answers the user question by searching for relevant rule texts, reading retrieved noisy rule texts (rule text 2 is irrelevant to the user question), interpreting the user scenario, and keeping asking follow-up questions to clarify the user's background until it concludes a final answer. Question-relevant requirements in the rule texts are bold.

the essential retrieval step in real scenarios. In the real world, a system needs to take the question, retrieve several relevant rule passages, read and reason over them to give the final decision. Therefore, it is promising to investigate an open-retrieval setting of conversational machine reading [63, 64, 18]. In the open-retrieval setting, the relevant rule texts are unknown, so a system

Prompt Question (Google search query): What's the most points scored in an NBA game?
Disambiguated QA Pairs:
Q$_1$: What's the most points scored in an NBA game by combined team? / **A**$_1$: 370
Q$_2$: What's the most points scored in an NBA game by a single team? / **A**$_2$: 186
Q$_3$: What's the most points scored in an NBA game by an individual? / **A**$_3$: 100

Relevant Wikipedia Page 1: The highest-scoring regular season game is the triple-overtime game between ... the two teams combined to score 370 points, with the pistons defeating the nuggets 186–184 ...
Relevant Wikipedia Page 2: Wilt Chamberlain scored an nba-record 100 points ...

Figure 8.2: An example from the AmbigQA [55] dataset. The **Prompt Question** is gathered from Google search queries and has three interpretations upon reading Wikipedia. **Disambiguated QA Pairs** are the full set of acceptable answers, paired with the disambiguated rewriting of the prompt question.

needs to retrieve question-relevant evidence from a collection of rule texts and answer users' high-level questions according to multiple retrieved rule texts in a conversational manner. Figure 8.1 gives an example for open-retrieval conversational machine reading. The user asks whether he is suitable for the loan program. The machine retrieves three question-relevant rule texts but only rule text 1 and rule text 3 are truly relevant. Then it understands the requirements listed in the rule text, finds the user meets "American small business" from the user scenario, asks two follow-up questions about "for-profit business" and "not get financing from other resources", and finally it concludes the answer "Yes" to the user's initial question.

Clarification QA Pair Generation for Ambiguous Questions. Open-domain Question Answering (QA) is the task of answering questions using a collection of passages with diverse topics [6, 23, 31]. Open-domain questions are highly likely to be ambiguous because people may not know about relevant topics when formulating them. For example, in Figure 8.2, the prompt question *"What's the most points scored in an NBA game?"* is ambiguous because the *score* in this question could be interpreted as *the combined score in a game* (Q_1A_1), *score from a single team* (Q_2A_2), or *score from an individual player* (Q_3A_3). According to a recent study, over 50% of Google search queries are ambiguous [55, 54]. Therefore, a system needs to adaptively predict a single answer or a set of equally plausible answers when the question has multiple interpretations. When a set of multiple answers is predicted, an unambiguous rewriting of the question that leads to each answer should also be provided to clarify each interpretation. Since more than 50% search queries are ambiguous, the future search engine may evolve into a conversational search engine that can ask you clarification questions before answering your query.

□ **End of chapter.**

Publications during Ph.D. Study

1. **Yifan Gao**, Chien-Sheng Wu, Jingjing Li, Shafiq Joty, Steven C.H. Hoi, Caiming Xiong, Irwin King, Michael R. Lyu, *Discern: Discourse-Aware Entailment Reasoning Network for Conversational Machine Reading*, in Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020), pages 2439–2449, November 16–20, 2020.

2. **Yifan Gao**, Piji Li, Wei Bi, Xiaojiang Liu, Michael R. Lyu, Irwin King, *Dialogue Generation on Infrequent Sentence Functions via Structured Meta-Learning*, in Findings of the Association for Computational Linguistics: EMNLP 2020, pages 431–440, November 16 - 20, 2020.

3. **Yifan Gao**, Chien-Sheng Wu, Shafiq Joty, Caiming Xiong, Richard Socher, Irwin King, Michael R. Lyu, Steven C.H. Hoi, *Explicit Memory Tracker with Coarse-to-Fine Reasoning for Conversational Machine Reading*, in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020), pages 935–945, July 5 - 10, 2020.

4. **Yifan Gao**, Piji Li, Irwin King, Michael R. Lyu, *Inter-*

connected Question Generation with Coreference Alignment and Conversation Flow Modeling, in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL 2019), pages 4853–4862, Florence, Italy, July 28 - August 2, 2019.

5. **Yifan Gao**, Lidong Bing, Wang Chen, Irwin King, Michael R. Lyu, *Difficulty Controllable Generation of Reading Comprehension Questions*, in Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI 2019). Main track. Pages 4968-4974, Macao, China, August 10-16, 2019.

6. **Yifan Gao**, Lidong Bing, Piji Li, Irwin King and Michael R. Lyu, *Generating Distractors for Reading Comprehension Questions from Real Examinations*, in Proceedings of the 33rd Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence (AAAI 2019). Main track. Pages 6423-6430, Honolulu, Hawaii, USA, January 27 - February 1, 2019.

7. Jingjing Li, **Yifan Gao**, Lidong Bing, Irwin King, Michael R. Lyu, *Improving Question Generation With to the Point Context*, in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP 2019), pages 3216–3226, Hong Kong, China, November 3–7, 2019.

8. Wang Chen, **Yifan Gao**, Jiani Zhang, Irwin King, Michael R. Lyu, *Title-Guided Encoding for Keyphrase Generation*, in Proceedings of the 33rd Association for the Advancement

of Artificial Intelligence (AAAI) Conference on Artificial Intelligence (AAAI 2019). Main track. Pages 6423-6430, Honolulu, Hawaii, USA, January 27 - February 1, 2019.

Note: The papers [1, 3, 4, 5, 6] are partially involved in this thesis.

Bibliography

[1] Jun Araki, Dheeraj Rajagopal, Sreecharan Sankaranarayanan, Susan Holm, Yukari Yamakawa, and Teruko Mitamura. Generating questions and multiple-choice answers using semantic analysis of texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1125–1136, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee.

[2] Lee Becker, Sumit Basu, and Lucy Vanderwende. Mind the gap: Learning to choose gaps for question generation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 742–751, Montréal, Canada, June 2012. Association for Computational Linguistics.

[3] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620, Doha, Qatar, October 2014. Association for Computational Linguistics.

[4] Eric Brill, Susan Dumais, and Michele Banko. An analysis of the AskMSR question-answering system. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 257–264. Association for Computational Linguistics, July 2002.

[5] Chia-Yin Chen, Hsien-Chin Liou, and Jason S. Chang. FAST – an automatic generation system for grammar tests. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 1–4, Sydney, Australia, July 2006. Association for Computational Linguistics.

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

[7] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[8] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural*

Language Processing, pages 2174–2184, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[9] Kevin Clark and Christopher D. Manning. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2256–2262, Austin, Texas, November 2016. Association for Computational Linguistics.

[10] Guy Danon and Mark Last. A syntactic approach to domain-specific automatic question generation. *CoRR*, abs/1712.09827, 2017.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[12] Li Dong and Mirella Lapata. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany, August 2016. Association for Computational Linguistics.

[13] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural

language understanding and generation. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 13042–13054, 2019.

[14] Xinya Du and Claire Cardie. Harvesting paragraph-level question-answer pairs from Wikipedia. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1907–1917, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[15] Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[16] Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. Applying deep learning to answer selection: A study and an open task. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2015, Scottsdale, AZ, USA, December 13-17, 2015*, pages 813–820. IEEE, 2015.

[17] David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager,

Nico Schlaefer, and Christopher A. Welty. Building watson: An overview of the deepqa project. *AI Mag.*, 31(3):59–79, 2010.

[18] Yifan Gao, Jingjing Li, Michael R. Lyu, and Irwin King. Open-retrieval conversational machine reading. *CoRR*, abs/2102.08633, 2021.

[19] Yifan Gao, Piji Li, Irwin King, and Michael R. Lyu. Interconnected question generation with coreference alignment and conversation flow modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4853–4862, Florence, Italy, July 2019. Association for Computational Linguistics.

[20] Yifan Gao, Chien-Sheng Wu, Shafiq Joty, Caiming Xiong, Richard Socher, Irwin King, Michael Lyu, and Steven C.H. Hoi. Explicit memory tracker with coarse-to-fine reasoning for conversational machine reading. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 935–945, Online, July 2020. Association for Computational Linguistics.

[21] H. C. Goodrich. Distractor efficiency in foreign language testing. *TESOL Quarterly*, 11:69, 1977.

[22] Qi Guo, Chinmay Kulkarni, Aniket Kittur, Jeffrey P. Bigham, and Emma Brunskill. Questimator: Generating knowledge assessments for arbitrary topics. In Subbarao Kambhampati, editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 3726–3732. IJCAI/AAAI Press, 2016.

[23] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. *International Conference on Machine Learning*, 2020.

[24] Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 221–231, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[25] Michael Heilman and Noah A. Smith. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617, Los Angeles, California, June 2010. Association for Computational Linguistics.

[26] Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. Tracking the world state with recurrent entity networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[28] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convo-

lutional neural networks and incremental parsing. To appear, 2017.

[29] Wenpeng Hu, Bing Liu, Jinwen Ma, Dongyan Zhao, and Rui Yan. Aspect-based question generation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018.

[30] Shafiq Joty, Giuseppe Carenini, and Raymond T. Ng. CODRA: A novel discriminative framework for rhetorical analysis. *Computational Linguistics*, 41(3):385–435, September 2015.

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

[32] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[33] Girish Kumar, Rafael Banchs, and Luis Fernando D'Haro. RevUP: Automatic gap-fill question generation from educational texts. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Ap-*

plications, pages 154–161, Denver, Colorado, June 2015. Association for Computational Linguistics.

[34] Igor Labutov, Sumit Basu, and Lucy Vanderwende. Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 889–898, Beijing, China, July 2015. Association for Computational Linguistics.

[35] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[36] Yunshi Lan and Jing Jiang. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 969–974, Online, July 2020. Association for Computational Linguistics.

[37] Carolin Lawrence, Bhushan Kotnis, and Mathias Niepert. Attending to future tokens for bidirectional sequence generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1–10, Hong Kong,

China, November 2019. Association for Computational Linguistics.

[38] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[39] Jing Li, Aixin Sun, and Shafiq R. Joty. Segbot: A generic neural text segmentation model with pointer network. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4166–4172. ijcai.org, 2018.

[40] Jiwei Li, Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1106–1115, Beijing, China, July 2015. Association for Computational Linguistics.

[41] Jiwei Li, Alexander H. Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. Learning through dialogue interactions by asking questions. In *5th International Conference on Learning Representations, ICLR 2017, Toulon,*

France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017.

[42] Chen Liang, Xiao Yang, Neisarg Dave, Drew Wham, Bart Pursel, and C. Lee Giles. Distractor generation for multiple choice questions using learning to rank. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 284–290, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[43] Chen Liang, Xiao Yang, Drew Wham, Bart Pursel, Rebecca Passonneaur, and C. Lee Giles. Distractor generation with generative adversarial nets for automatically creating fill-in-the-blank questions. In Óscar Corcho, Krzysztof Janowicz, Giuseppe Rizzo, Ilaria Tiddi, and Daniel Garijo, editors, *Proceedings of the Knowledge Capture Conference, K-CAP 2017, Austin, TX, USA, December 4-6, 2017*, pages 33:1–33:4. ACM, 2017.

[44] Yi Liao, Lidong Bing, Piji Li, Shuming Shi, Wai Lam, and Tong Zhang. QuaSE: Sequence editing under quantifiable guidance. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3855–3864, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[45] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[46] Xiang Lin, Shafiq Joty, Prathyusha Jwalapuram, and M Saiful Bari. A unified linear-time framework for sentence-level discourse parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4190–4200, Florence, Italy, July 2019. Association for Computational Linguistics.

[47] David Lindberg, Fred Popowich, John Nesbit, and Phil Winne. Generating natural language questions to support learning on-line. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 105–114, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[48] Jeffrey Ling and Alexander Rush. Coarse-to-fine attention models for document summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 33–42, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[49] 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 preprint arXiv:1907.11692*, 2019.

[50] Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Zhu. Knowledge base question answering via encoding of complex query graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2185–2194, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[51] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[52] William C Mann and Sandra A Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text-interdisciplinary Journal for the Study of Discourse*, 8(3):243–281, 1988.

[53] Karen Mazidi and Rodney D. Nielsen. Linguistic considerations in automatic question generation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 321–326, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[54] Sewon Min, Kenton Lee, Ming-Wei Chang, Kristina Toutanova, and Hannaneh Hajishirzi. Joint passage ranking for diverse multi-answer retrieval. 2021.

[55] Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. AmbigQA: Answering ambiguous open-domain questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5783–5797, Online, November 2020. Association for Computational Linguistics.

[56] Ruslan Mitkov and Le An Ha. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applica-*

tions Using Natural Language Processing, pages 17–22, 2003.

[57] Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Roxana Girju, Richard Goodrum, and Vasile Rus. The structure and performance of an open-domain question answering system. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 563–570, Hong Kong, October 2000. Association for Computational Linguistics.

[58] Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1802–1813, Berlin, Germany, August 2016. Association for Computational Linguistics.

[59] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[60] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas, November 2016. Association for Computational Linguistics.

[61] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[62] Xipeng Qiu and Xuanjing Huang. Convolutional neural tensor network architecture for community-based question answering. In Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1305–1311. AAAI Press, 2015.

[63] Chen Qu, Liu Yang, Cen Chen, W. Bruce Croft, Kalpesh Krishna, and Mohit Iyyer. Weakly-supervised open-retrieval conversational question answering. In Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani, editors, *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part I*, volume 12656 of *Lecture Notes in Computer Science*, pages 529–543. Springer, 2021.

[64] Chen Qu, Liu Yang, Cen Chen, Minghui Qiu, W. Bruce Croft, and Mohit Iyyer. Open-retrieval conversational question answering. In Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu, editors, *Proceedings of the 43rd International ACM SIGIR conference on research and development in*

Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020, pages 539–548. ACM, 2020.

[65] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

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

[67] Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, March 2019.

[68] Siva Reddy, Mirella Lapata, and Mark Steedman. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392, 2014.

[69] Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. Interpretation of natural language rules in conversational machine reading. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2087–2097, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[70] Keisuke Sakaguchi, Yuki Arase, and Mamoru Komachi. Discriminative approach to fill-in-the-blank quiz generation for language learners. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 238–242, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[71] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[72] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[73] Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. Generating factoid questions with recurrent neural networks: The 30M factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 588–598, Berlin, Germany, August 2016. Association for Computational Linguistics.

[74] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi S. Jaakkola. Style transfer from non-parallel text by cross-alignment. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6830–6841, 2017.

[75] Robert F. Simmons. Answering english questions by computer: a survey. *Commun. ACM*, 8(1):53–70, 1965.

[76] Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 569–574, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[77] Katherine Stasaski and Marti A. Hearst. Multiple choice question generation utilizing an ontology. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 303–312, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[78] Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gür, Zenghui Yan, and Xifeng Yan. On generating characteristic-rich question sets for QA evaluation. In

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 562–572, Austin, Texas, November 2016. Association for Computational Linguistics.

[79] Saku Sugawara, Yusuke Kido, Hikaru Yokono, and Akiko Aizawa. Evaluation metrics for machine reading comprehension: Prerequisite skills and readability. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 806–817, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[80] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[81] Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[82] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D.

Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014.

[83] Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. oLMpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics*, 8:743–758, 2020.

[84] Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. From neural sentence summarization to headline generation: A coarse-to-fine approach. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 4109–4115. ijcai.org, 2017.

[85] Ming Tan, Bing Xiang, and Bowen Zhou. Lstm-based deep learning models for non-factoid answer selection. *CoRR*, abs/1511.04108, 2015.

[86] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Multi-range reasoning for machine comprehension. *CoRR*, abs/1803.09074, 2018.

[87] Lucy Vanderwende. Answering and questioning for machine reading. In *Machine Reading, Papers from the 2007 AAAI Spring Symposium, Technical Report SS-07-06, Stanford, California, USA, March 26-28, 2007*, page 91. AAAI, 2007.

[88] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser,

and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[89] Nikhil Verma, Abhishek Sharma, Dhiraj Madan, Danish Contractor, Harshit Kumar, and Sachindra Joshi. Neural conversational QA: Learning to reason vs exploiting patterns. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7263–7269, Online, November 2020. Association for Computational Linguistics.

[90] Shuohang Wang, Mo Yu, Jing Jiang, and Shiyu Chang. A co-matching model for multi-choice reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 746–751, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[91] Siyuan Wang, Zhongyu Wei, Zhihao Fan, Yang Liu, and Xuanjing Huang. A multi-agent communication framework for question-worthy phrase extraction and question generation. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu,*

Hawaii, USA, January 27 - February 1, 2019, pages 7168–7175. AAAI Press, 2019.

[92] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[93] Yansen Wang, Chenyi Liu, Minlie Huang, and Liqiang Nie. Learning to ask questions in open-domain conversational systems with typed decoders. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2193–2203, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[94] Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[95] Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*,

pages 438–449, Valencia, Spain, April 2017. Association for Computational Linguistics.

[96] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019.

[97] Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819, Florence, Italy, July 2019. Association for Computational Linguistics.

[98] Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. Exploiting rich syntactic information for semantic parsing with graph-to-sequence model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 918–924, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[99] Yichong Xu, J. Liu, Jianfeng Gao, Y. Shen, and Xiaodong Liu. Dynamic fusion networks for machine reading comprehension. *arXiv: Computation and Language*, 2017.

[100] Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William Cohen. Semi-supervised QA with generative domain-adaptive nets. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers), pages 1040–1050, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[101] Mark Yatskar. A qualitative comparison of CoQA, SQuAD 2.0 and QuAC. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2318–2323, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[102] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China, July 2015. Association for Computational Linguistics.

[103] 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 *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[104] Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Saizheng Zhang, Sandeep Subramanian, and Adam Trischler. Machine comprehension

by text-to-text neural question generation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 15–25, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[105] Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[106] Victor Zhong, Caiming Xiong, and Richard Socher. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1458–1467, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[107] Victor Zhong and Luke Zettlemoyer. E3: Entailment-driven extracting and editing for conversational machine reading. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2310–2320, Florence, Italy, July 2019. Association for Computational Linguistics.

[108] Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. Neural question generation from text: A preliminary study. In Xuanjing Huang, Jing Jiang, Dongyan Zhao, Yansong Feng, and Yu Hong, editors, *Natural Language Processing and Chinese Computing -*

6th CCF International Conference, NLPCC 2017, Dalian, China, November 8-12, 2017, Proceedings, volume 10619 of *Lecture Notes in Computer Science*, pages 662–671. Springer, 2017.

[109] Haichao Zhu, Furu Wei, Bing Qin, and Ting Liu. Hierarchical attention flow for multiple-choice reading comprehension. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6077–6085. AAAI Press, 2018.