University of California
Santa Barbara

Accessing Diverse Web Knowledge with Natural Language Interface

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

Wenhu Chen

Committee in charge:

Professor William Yang Wang, Chair
Professor Xifeng Yan, Co-Chair
Professor Ambuj Singh

June 2021

The Dissertation of Wenhu Chen is approved.

Professor Xifeng Yan, Co-Chair

Professor Ambuj Singh

Professor William Yang Wang, Committee Chair

June 2021

Accessing Diverse Web Knowledge with Natural Language Interface

by

Wenhu Chen

I dedicate this thesis to my family for nursing me with affection and love and their dedicated partnership in my life

Acknowledgements

I owe deep gratitude to my advisors William Yang Wang and Xifeng Yan. Without their guidance, this dissertation is not possible. They brought me to UCSB and is always supportive in both work and life. William and Xifeng led me to the emerging research area of natural language processing and has provided endless support in terms of research, career choice, etc. My Ph.D. journey was a very pleasant experience because of them. Besides, I thank my additional dissertation committee member Ambuj Singh for their insightful feedback and comments.

Throughout my graduate school, I am very thankful to all my friends, colleagues, and collaborators for insightful discussion about my work and career choice. Here I want to thank William Cohen, Ming-Wei Chang, Eva Schlinger, Livio Baldini Soares, Zhe Gan, Yu Cheng, Jingjing Liu, Yilin Shen, Yu Su, Jianshu Chen, Dong Yu, Eric Xin Wang, Wenhan Xiong, Zhiyu Chen, Liangming Pan, Hanwen Zha, Jingzhou Liu, Shiyang Li, Hong Wang, Xinyi Wang, Yi-Lin Tuan, Lily Ou, Rohan Bhatia, Pengda Qin, Vardaan Pahuja, Tao Yu, Jiacheng Xu, Yuchen Lin, Guanlin Li, Shuo Ren, Shujie Liu.

Finally, my greatest gratitude goes to my parents for their unconditional love and support. I will always make you proud!

Thesis Statement

I aim to build natural language interface with machine learning techniques to empower humans to access the massive amount of knowledge on the Web, which exists in various forms like text, graphs, tables, images, videos. My ultimate goal is design algorithms to enable machines to think and reason like human while generalizing to large scale in a highly effective manner.

Abstract

Accessing Diverse Web Knowledge with Natural Language Interface

by

Wenhu Chen

The Web is the primary source for humans to access information in our daily life. In the Information Age, Web knowledge has exploded and distributed in diverse forms like text, tables, charts, graphs, images. To help humans cope with such massive amount of Web information, building an intelligent Natural Language Interface has become a primary interest in artificial intelligence. The natural language interface needs to understand the human input and ground on Web knowledge to provide useful information. Technically, the interface needs two components: 1) natural language understanding: understanding the semantics of the natural language to navigate to supportive evidence. 2) natural language generation: grounding on the supportive evidence to generate natural language response.

In the first part, we will focus on the understanding component and use question answering as our evaluation task. Specifically, I will cover three different grounding scenarios: 1) how to ground on the given hybrid data (structured + unstructured) to answer multi-hop questions, 2) how to search over the Web and then ground on retrieved hybrid data to answer multi-hop questions 3) how to ground on visual data to answer compositional questions.

In the second part, we will focus on the generation component and use data-to-text generation as our evaluation task. Specifically, I will discuss two different scenarios: 1) how to ground on structured tabular data to generate logically entailed claims, 2) how to leverage unlabeled Web data for pre-training to improve existing data-to-text models.

In these scenarios, we designed novel algorithms to dramatically improve the existing models' capability to generate more consistent and coherent text.

Finally, we summarize the strengths, weaknesses, and implications of our work, and discuss the future research plan of pushing the direction forward.

Contents

Chapter 1

Introduction

1.1 Natural Language Interface

Web is the primary source for humans to search and gain information in daily life. With the recent trend of digitization, the Web plays a more important role than ever. However, the information stored on the Web has also exploded and increases in an exponential manner. With such a vast amount of information on the Web, it's almost unavoidable to build a smart and intelligent interface to help humans access these information. Such interfaces can take human queries as the input to search over the Web and ground on the supportive evidence to communicate with humans. These interfaces can smartly understand the humans' intention to perform an automated information-seeking process to cater knowledge in a human-understandable manner. The primary goal of this thesis is to build such an intelligent natural language interface to meet humans their daily needs.

Specifically, the interface model should have two capabilities: (1) NLU: understanding human language (i.e. taking human language as input) and ground on web knowledge to make a find supporting evidence, (2) NLG: generating human language by grounding

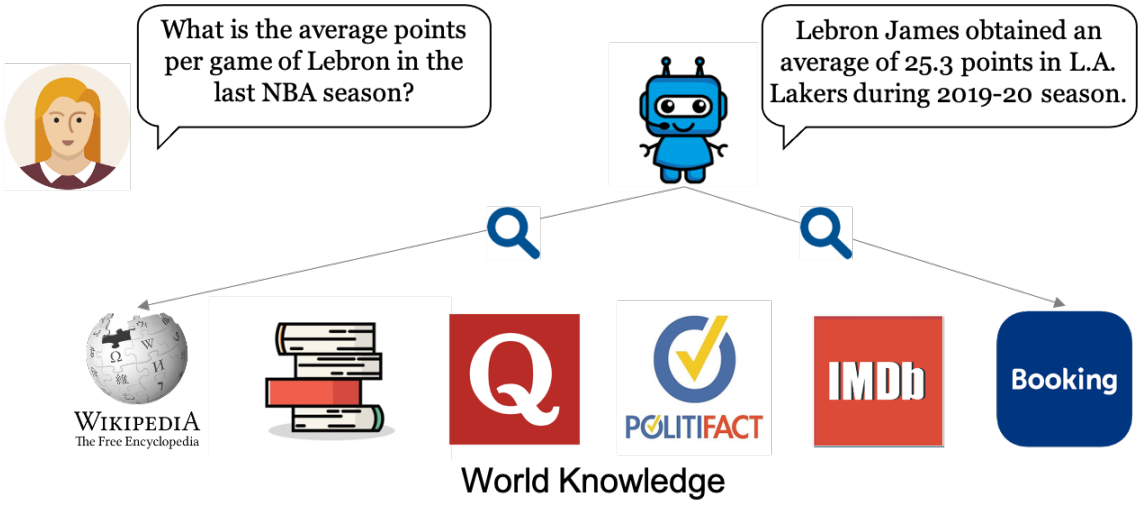

Figure 1.1: Natural language interface to access web knowledge.

on the supportive evidence to communicate with humans.

A standard example for such a natural language interface is Google Siri or Amazon Alexa, which can take voice input to help customers accomplish specific tasks in an interactive manner. As depicted in Figure 1.1, the human wants to know the specific information about sports, the interface needs to traverse the web and find the supportive evidence to generate natural language response.

In Figure 1.2, the agent will understand the user query's purpose, for example, who is "Lebron". There are tens of thousands of "Lebron" in the world. However, by combining with the linguistic context in the query. We can accurately navigate to "Lebron James", the professional basketball player in L.A. Lakers in NBA. After understanding the semantics, are we able to truly understand the query's semantic form as "SELECT PPG FROM TABLE Lebron-Career-Statistics WHERE Year=2019-2020". From this example, we show that human language is highly ambiguous, containing diverse, metaphoric expressions, which poses great challenges to the machine learning models. Furthermore, natural language is highly contextualized, which means that understanding human queries normally requires very rich external/background world knowledge,

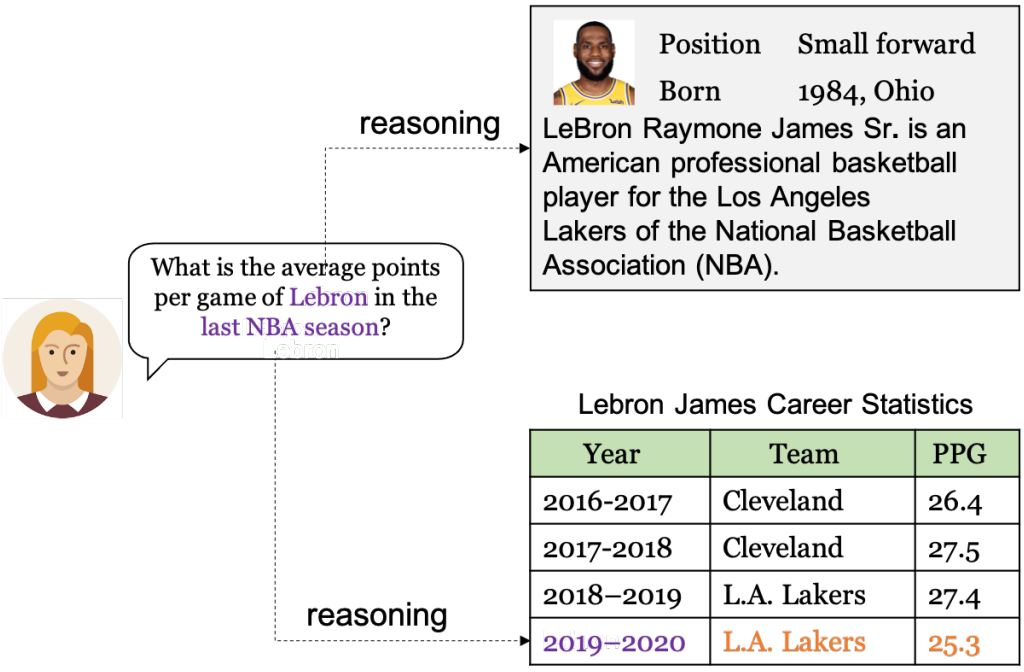

Figure 1.2: Natural language understanding.

which goes beyond their lexical forms. Therefore, how can we leverage the external or contextualized knowledge to understand the semantics of the natural language inputs in the interface is of great interest to us, which is also the primary focus of this thesis.

After reasoning and understanding the semantics, we will derive the supportive evidence "PPG=25.3, Team=L.A. Lakers". We need to ground on this supportive evidence to generate a natural language response. As described in Figure 1.3, the model needs to understand and then convert the structured knowledge into coherent a natural language sentence. This generation step is also a challenging task in the sense that the generation should not only remain faithful to the world knowledge but also adhere to the linguistic rules and patterns. However, in order to measure or quantify these aspects, we need perfect understanding models, which we apparently don't have. Thus, understanding and generation can be seen as chicken-egg problems, which we need to solve jointly.

In this thesis, I will break the natural language interface into NLU and NLG aspects

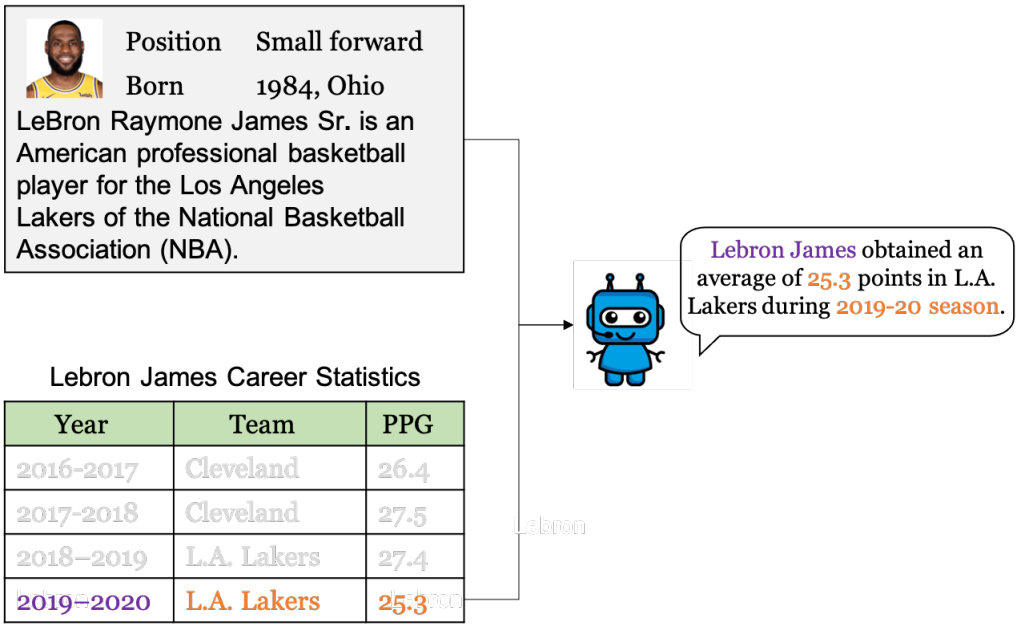

Figure 1.3: Natural language generation.

and talk about them separately.

1.2 Diverse Web Knowledge

Instead of operating on knowledge sources with a strict schema such as a database, we propose to operate over the raw Web, which contains a large amount of up-to-date open-domain information (high BREADTH), as the knowledge source. An important characteristic of the Web is that the information is represented in diverse forms, containing unstructured textual data, semi-structured tabular data, and visual data, etc. One of the key challenges is how to handle these diverse forms of data, for example, how to leverage the evidence distributed in different forms and combine them together to perform compositional reasoning. As depicted in Figure 1.2, the information for answering the given query is distributed in both the "Lebron James" passage, and the "Lebron James Career Statistics" table. To empower the natural language interface to handle broader

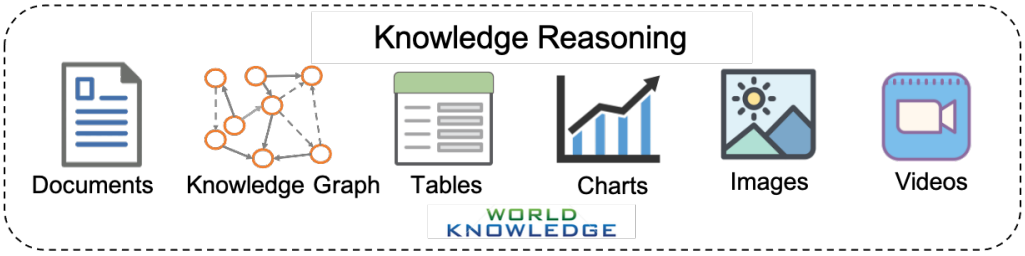

Figure 1.4: Natural language generation.

situations in real-world applications, the capability to reason across various forms including passages, graphs, tables, charts, images are essential. In this thesis, we are specifically interested in answering the following two research questions:

1. How can we ground on diverse forms of knowledge to understand human language?

2. How can we ground on diverse forms of knowledge to generate human language?

In the Part I, we will discuss natural language understanding under different scenarios, with tables, text, and visual data. We will focus on developing algorithms to integrate structured and unstructured data for information integration. In the Part II, we will discuss natural language generation over three different knowledge sources including tables, knowledge graphs, and dialog acts. The focus is to enhance the generation model's faithfulness, logical consistency, and controllability over generated sentences.

Finally, we summarize the strengths, weaknesses, and implications of our work, and discuss the future research plan of pushing the direction forward.

1.3 Evaluation

1.3.1 NLU with Question Answering

Natural language understanding (NLU) can be formulated as learning the deep semantics of the purpose of the given natural language. It consists of the capability of a

program system to translate sentences into an internal representation, which could be understood by humans and machines. However, the internal "meaning" representation could be quite vague and not very well defined under many scenarios, which poses great challenges for us to fairly and accurately evaluate the NLU model's performance. Therefore, the existing community mostly resorts to different tasks like question answering, sentiment analysis, natural language inference, state tracking to evaluate the model's performance in understanding human language. In this thesis, we will mainly use question answering [1] as the main task to evaluate the model's capability to perform natural language understanding. Here, we compare the model's predicted answer against the human-annotated answer to estimate whether the model truly understands the semantics of the given natural language questions. Such an approach has been quite widely used in different benchmarks like GLUE [2], SUPERGLUE [3], etc. Specifically, we use the exact match and F1 scores as the indicator to calculate the overlap of model prediction against the human reference.

1.3.2 NLG with Data-to-Text Generation

Natural language generation (NLG) can be formulated as generating a coherent sentence given the meaning representation. It consists of the capability of translating the internal representation into a human understandable sentence. However, the internal representation could also be hard to define. Thus the existing community mostly resorts to different downstream tasks (i.e. data-to-text generation, which grounds on the given structured to generate natural language) to evaluate the model's performance in generating human language. In this thesis, we will mainly use data-to-text generation [4] as the main task to evaluate the model's capability to perform natural language generation. Here, we compare the model's generated sentence against human-annotated sentence to

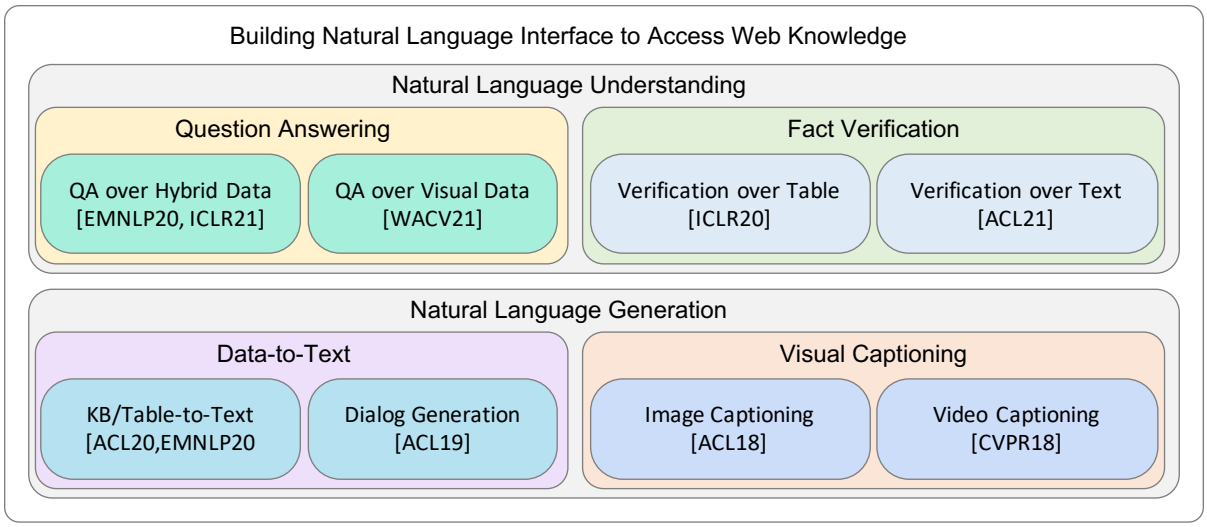

Figure 1.5: My contribution to Natural Language Interface.

estimate whether the model truly generates faithful natural language sentences. Specifically, we use different metrics like BLEU [5], METEROR [6], ROUGE [7], other proposed faithfulness metrics and even human evaluation to estimate model's generation capability.

1.4 Contributions

Here I summarize my contributions to building natural language interface in Figure 1.5. Throughout my Ph.D. career, I have been investigating the two aspects:

- Understanding human language with external Web knowledge, specifically, I worked on question answering and fact verification. In [8, 9], I investigated how to leverage hybrid structured and unstructured data to build a QA system. In [10], I studied how to leverage meta-learning to enhance neural module networks to answer visual questions. In [11, 12], I investigated how to utilize tables and text to verify misinformation.

- Generating human-like language from external Web knowledge, specifically, I worked

on data-to-text generation and visual captioning. In [13, 14], I investigated how to convert structured data into verbal form. In [15], I studied how to generate response conditioned on conversation history. In [16, 17], I studied how to describe visual content like images or videos with natural language descriptions.

With these efforts, I aim to build the next-generation natural language interface that can cater Web information to humans to help them accomplish their everyday needs.

Part I

Natural Language Understanding over Diverse Web Knowledge

Chapter 2

Question Answering over Hybrid Web Knowledge

2.1 Introduction

Existing question answering frameworks focus mostly on dealing with homogeneous information, consisting of either text [1, 18, 19] or knowledge graphs [20, 21, 22, 23], or tables [24, 25, 26] alone. However, as the human knowledge is generally distributed across different forms on the Web due to their different properties. It might be insufficient to incorporate only a single data form for building question answering systems. Therefore, it is our primary interest to build hybrid models to aggregate information from different forms from the Web. There has been some pioneering work on building hybrid QA systems [27, 28, 29]. These methods adopts KB-only datasets [20, 21, 23] to simulate a hybrid setting by randomly masking KB triples and replace them with text corpus. Experimental results have proved decent improvement, which shed lights on the potential of hybrid question answering systems to integrate heterogeneous information.

Though there already exist numerous valuable questions answering datasets as listed

Dataset	Size	#Documents	KB/Table	Multi-Hop
WebQuestions	5.8K	no	yes	yes
WebQuestions-Simple	4.7K	no	yes	yes
WebQuestions-Complex	34K	no	yes	yes
MetaQA	400k	no	yes	yes
WikiTableQA	22K	no	yes	yes
SQuAD-v1	107K	1	no	no
DROP	99K	1	no	yes
TriviaQA	95K	>1	no	no
HotpotQA	112K	>1	no	yes
NaturalQuestion	300K	>1	no	yes
HybridQA	70K	>>1	yes	yes

Table 2.1: Comparison of existing datasets, where #docs means the number of documents provided for a specific question. 1) KB-only datasets: WebQuestions [20], WebQSP [22], WebComplex [23], MetaQA [30], WikiTableQuestion [24]. 2) Text-only datasets with single passage: like SQuAD [1], DROP [31]. 3) open-domain Text-Only dataset: TriviaQA [19], HotpotQA [32], Natural Questions [33].

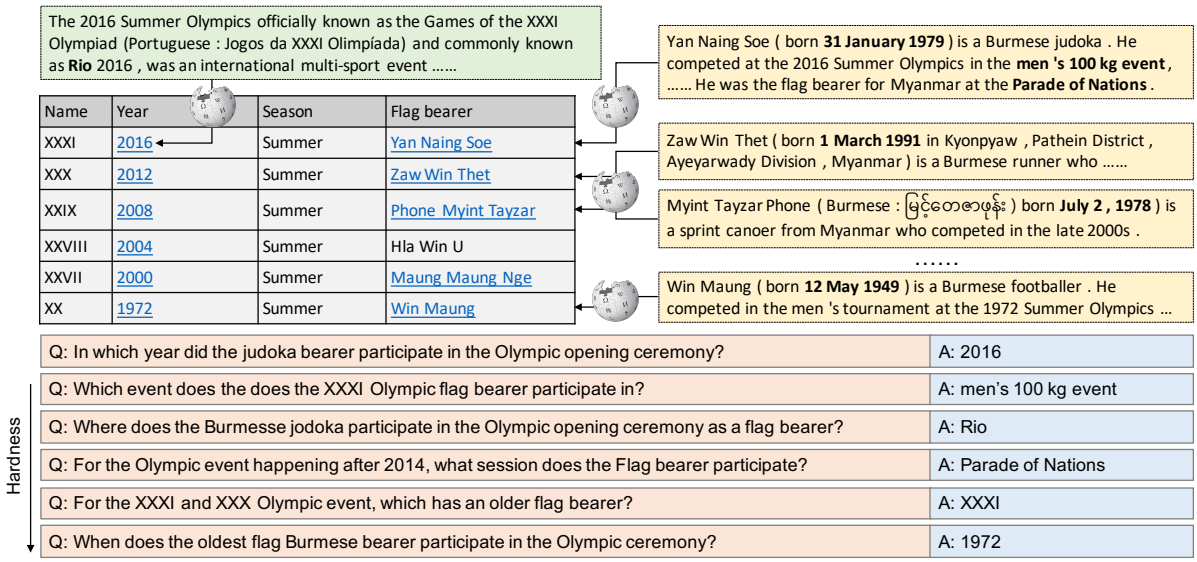

Figure 2.1: Examples of annotated question answering pairs from Wikipedia page. Underlined entities have hyperlinked passages, which are displayed in the boxes. The lower part shows the human-annotated question-answer pairs roughly categorized based on their hardness.

in Table 2.1, these datasets were initially annotated to use either structured or unstructured information during annotation. There is no incentive for the models to aggregate heterogeneous information to find the answer. Therefore, designing hybrid question answering systems would probably yield very marginal benefits over the non-hybrid ones, which greatly hinders the research development in building hybrid question answering systems.

To fill in the gap, we propose to construct a novel heterogeneous QA dataset HybridQA, which is collected by crowdsourcing based on Wikipedia tables. During annotation, each crowd worker is presented with a Wikipedia table along with its hyperlinked Wikipedia passages to annotate questions requiring aggregating information from both forms. The dataset consists of roughly 70K question-answering pairs aligned with roughly 13K Wikipedia tables, roughly 5 questions/table. As the tables presented on Wikipedia [34] were curated from high-standard professionals to organize homogeneous information, its coverage is mostly absent in the accompanying text. Such a complementary nature makes WikiTables an ideal environment for hybrid question answering. To ensure that the answers cannot be hacked by single-hop or homogeneous models, we carefully employ different strategies to calibrate the annotation process. An example is demonstrated in Figure 2.1. This table is aimed to describe Burmese flag bearers over different Olympic events, where the second column has hyperlinked passages about the Olympic event, and the fourth column has hyperlinked passages about biography individual bearers. The dataset is both multi-hop and hybrid in the following senses: 1) the question requires multiple hops to achieve the answer, each reasoning hop may utilize either tabular or textual information. 2) the answer may come from either the table or a passage.

In our experiments, we implement three models, namely Table-only model, Passage-only, and a heterogeneous model HYBRIDER, which combines both information forms

#Table	#Row × #Column	#Total Cell	#Links/Table
13,000	15.7 × 4.4	70	44
#Passage	#Words/Passage	#Question	#Words/Question
293,269	103	69,611	18.9

Table 2.2: Statistics of Table and Passage in our dataset.

to perform multi-hop reasoning. Our Experiments show that two homogeneous models only achieve EM lower than 20%, while HYBRIDER can achieve an EM over 40%, which concludes the necessity to do multi-hop reasoning over heterogeneous information on HybridQA. As the HYBRIDER is still far behind human performance, we believe that it would be a challenging next-problem for the community.

2.2 Dataset

In this section, we describe how we crawl the high-quality tables with their associated passages automatically, and then describe how we collect hybrid questions. The statistics of HybridQA is shown in Table 2.2.

Table/Passage Collection To ease the annotation burden, we apply the following rules during table crawling. 1) we need tables with rows between 5-20, columns between 3-6, which is appropriate for the crowd-workers to visualize. 2) we ensure all the tables to have hyperlinked cells over 35% of its total cells, which can provide an abundant amount of textual information. For each hyperlink in the table, we retrieve its Wikipedia page and truncate at most the first 12 sentences from its introduction session as the associated passage. 3) we apply some additional rules to avoid improper tables and finally collect 13,000 high-quality tables.

Question/Answer Collection We release 13K HITs (human intelligence task) on the Amazon Mechanical Turk platform, where each HIT presents the crowd-worker with one crawled Wikipedia table along with its hyperlinked passages. We require the worker to write six questions as well as their answers. The question annotation phase is not trivial, as we specifically need questions that rely on both tabular and textual information. In order to achieve that, we exemplify abundant examples in our Amazon Turker interface with detailed explanations to help crowd-workers to understand the essence of the "hybrid" question. The guidelines are described as follows:

- The question requires at least 2 steps of reasoning over the two information forms of to derive the answer.
- Table reasoning step specifically includes (i) filter our table rows based on equal or greater or less, e.g. "For the XXXI Olympic event", (ii)) superlative operation over a column, e.g. "the earliest Olympic event", (iii) hop between two cells, e.g. "Which event ... participate in ...", (iv) extract information from table, e.g. "In which year did the player ... ".
- Text reasoning step specifically includes (i) select passages based on the certain mentions, e.g. "the judoka bearer", (ii) extract a span from the passage as the answer. (iii) compare numeric values in the passages.
- The answer should be a minimum text span from either a table cell or a specific passage, which can be used to answer the question.

Based on the above criteria, we hire five CS-majored graduate students as our "human expert" to decide the acceptance of a HIT. The average completion time for one HIT is 12 minutes, and payment is $2.3 U.S. dollars/HIT.

Annotation De-biasing As has been suggested in previous papers [35, 36, 37], the existing benchmarks on multi-hop reasoning question answering have annotation biases, which makes designing multi-hop models unnecessary. We discuss different biases and our prevention as follows:

- *Table Bias*: our preliminary study observes that the annotators prefer to ask questions regarding the top part of the table. In order to deal with this issue, we explicitly highlight certain regions in the table to encourage crowd-workers to raise questions regarding the given uniformly-distributed regions.

- *Passage Bias*: the preliminary study shows that the annotators like to ask questions regarding the first few sentences in the passage. In order to deal with such a bias, we use an algorithm to match the answer with linked passages to find their span and reject the HITs, which have all the answers centered around the first few sentences.

- *Question Bias*: the most difficult bias to deal with is the "fake" hybrid question like "when is 2012 Olympic Burmese runner flag bearer born?" for the table listed in Figure 2.1. Though it seems that "2012 Olympic" is needed to perform hop operation on the table, the "runner flag bearer" already reviews the bearer as "Zaw Win Thet" because there is no other runner bearer in the table. With that said, reading the passage of "Zaw Win Thet" alone can simply lead to the answer. In order to cope with such a bias, we ask "human experts" to spot such questions and reject them.

Statistics After we harvest the human annotations from 13K HITs (78K questions), we trace back the answers to its source (table or passage). Then we apply several rules to further filter out low-quality annotations: 1) the answer cannot be found from either table or passage, 2) the answer is longer than 20 words, 3) using a TF-IDF retriever

can directly find the answer passage with high similarity without relying on tabular information.

We filter the question-answer pairs based on the previous criteria and release the filtered version. As our goal is to solve multi-hop hybrid questions requiring a deeper understanding of heterogeneous information. We follow HotpotQA [32] to construct a more challenging dev/test split in our benchmark. Specifically, we use some statistical features like the "size of the table", "similarity between answer passage and question", "whether question directly mentions the field", etc. to roughly classify the question into two difficulty levels: simple (65%) and hard (35%). We construct our dev and test set by sampling half-half from the two categories. We match the answer span against all the cells and passages in the table and divide the answer source into three categories: 1) the answer comes from a text span in a table cell, 2) the answer comes from a certain linked passage, 3) the answer is computed by using numerical operation like 'count', 'add', 'average', etc. The matching process is approximated, not guaranteed to be 100% correct. We summarize our findings in Table 2.3. In the following experiments, we will report the EM/F1 score for these fine-grained question types to better understand our results.

Split	Train	Dev	Test	Total
In-Passage	35,215	2,025	20,45	39,285 (56.4%)
In-Table	26,803	1,349	1,346	29,498 (42.3%)
Computed	664	92	72	828 (1.1%)
Total	62,682	3,466	3,463	69,611 (100%)

Table 2.3: Data Split: In-Table means the answer comes from plain text in the table, and In-Passage means the answer comes from certain passage.

2.3 Data Analysis

In this section, we specifically analyze the different aspects of the dataset to provide the overall characteristics of the new dataset.

2.3.1 Question Types

We heuristically identified question types for each collected question. To identify the question type, we locate the central question word (CQW) in the question and take the neighboring three tokens [32] to determine the question types. We visualize the distribution in Figure 2.2, which demonstrates the syntactic diversity of the questions in HybridQA.

Figure 2.2: The type of questions in HybridQA, question types are extracted using rules starting at the question words or preposition before them.

2.3.2 Answer Types

We further sample 100 examples from the dataset and present the types of answers in Table 2.4. As can be seen, it covers a wide range of answer types. Compared to [32], our dataset covers more number-related or date-related questions, which reflects the nature of tabular data.

Answer Type	%	Example(s)
Location	22	Balestier Road, Atlanta
Number	22	762 metres (2,500 ft), 15,000
Date	20	April 25 , 1994, 1913
Person	15	Bärbel Wöckel, Jerry
Group	3	Hallmark Entertainment
Event	3	Battle of Hlobane
Artwork	1	Songmaster
Adjective	4	second-busiest
Other proper noun	8	Space Opera, CR 131
Common noun	1	other musicians

Table 2.4: Types of answers in HybridQA.

2.3.3 Inference Types

We analyze multi-hop reasoning types in Figure 2.3. According to our statistics, most of the questions require two or three hops to find the answer.

1. Type I question (23.4%) uses Table → Passage chain, it first uses table-wise operations (equal/greater/less/first/last/argmax/argmin) to locate certain cells in the table, and then hop to their neighboring hyperlinked cells within the same row, finally extracts a text span from the passage of the hyperlinked cell as the answer.

2. Type II question (20.3%) uses Passage → Table chain, it first uses cues present in the question to retrieve related passage, which traces back to certain hyperlinked

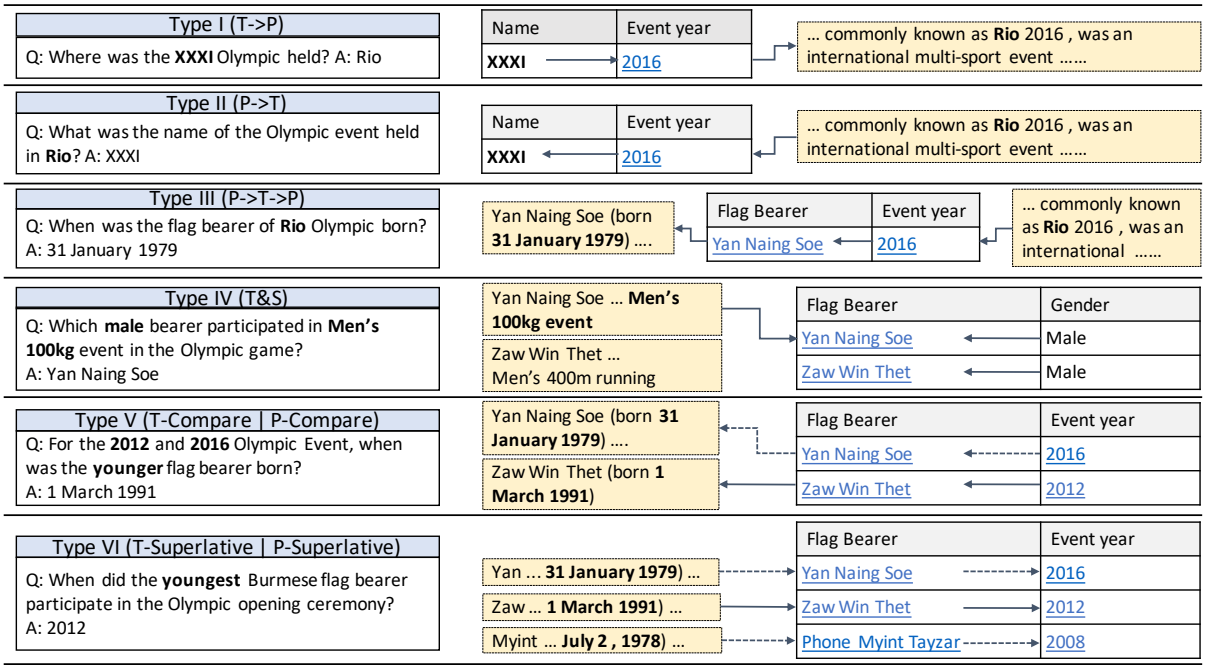

Figure 2.3: Illustration of different types of multi-hop questions.

cells in the table, and then hop to a neighboring cell within the same row, finally extracts text span from that cell.

3. Type III question (35.1%) uses Passage →Table→Passage chain, it follows the same pattern as Type II, but in the last step, it hops to a hyperlinked cell and extracts answer from its linked passage. This is the most common pattern.

4. Type IV question (17.3%) uses Passage and Table jointly to identify a hyperlinked cell based on table operations and passage similarity and then extract the plain text from that cell as the answer.

5. Type V question (3.1%) involves two parallel reasoning chain, while the comparison is involved in the intermediate step to find the answer.

6. Type VI questions (0.8%) involve multiple reasoning chains, while superlative in

involved in the intermediate step to obtain the correct answer.

2.4 Model

In this section, we propose three models we use to perform question answering on HybridQA.

2.4.1 Table-Only Model

In this setting, we design a model that can only rely on the tabular information to find the answer. Our model is based on the SQL semantic parser [25, 38], which uses a neural network to parse the given questions into a symbolic form and execute against the table. We follow the SQLNet [38] to flatten the prediction of the whole SQL query into a slot filling procedure. More specifically, our parser model first encode the input question q using BERT [39] and then decode the `aggregation`, `target`, `condition` separately as described in Figure 2.4. The `aggregation` slot can have the following values of "argmax, argmin, argmax-date, argmin-date", the `target` and `condition` slots have their potential values based on the table field and its corresponding entries. Though we do not have the ground-truth annotation for these simple SQL queries, we can use heuristics to infer them from the denotation. We use the synthesized question-SQL pairs to train the parser model.

2.4.2 Passage-Only Model

In this setting, we design a model that only uses the hyperlinked passages from the given table to find the answer. Our model is based on DrQA [40], which first uses an ensemble of several retrievers to retrieve related documents and then concatenate

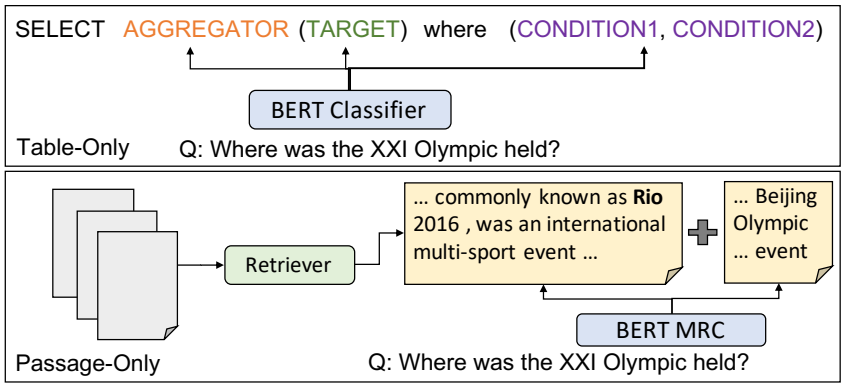

Figure 2.4: Illustration of the table-only and passage-only baselines, both are based on BERT Encoder.

several documents together to do reading comprehension with the state-of-the-art BERT model [39]. The basic architecture is depicted in Figure 2.4, where we use the retriever to retrieve the top-5 passages from the pool and then concatenate them as a document for the MRC model, and the maximum length of the concatenated document is set to 512.

2.4.3 hybrider

In order to cope with heterogeneous information, we propose a novel architecture called HYBRIDER. We divide the model into two phases as depicted in Figure 2.6 and describe them separately below:

Linking This phase is aimed to link questions to their related cells from two sources:

- Cell Matching: it aims to link cells explicitly mentioned by the question. The linking consists of three criteria, 1) the cell's value is explicitly mentioned, 2) the cell's value is greater/less than the mentioned value in question, 3) the cell's value is maximum/minimum over the whole column if the question involves superlative words.

- Passage Retriever: it aims to link cells implicitly mentioned by the question through its hyperlinked passage. The linking model consists of a TD-IDF retriever with 2-3 gram lexicon and a longest-substring retriever, this ensemble retriever calculates the distances with all the passages in the pool and highlight the ones with cosine distance lower than a threshold τ. The retrieved passages are mapped back to the linked cell in the table.

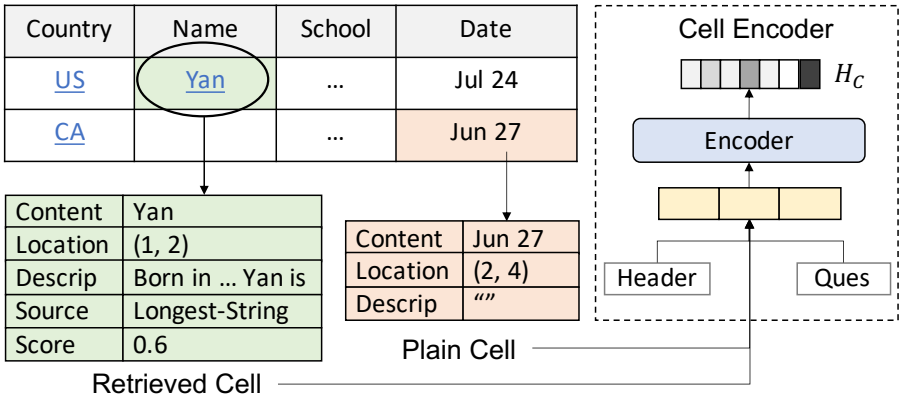

Figure 2.5: Illustration of cell encoder of retrieved (green) and plain cells (orange).

We call the set of cells from these two sources as "retrieved cells" denotes by $\mathcal{C}$. Each retrieved cell c is encoded by 5-element tuple `(content, location, description, source, score)`. `Content` represents the string representation in the table, `Content` refers to the absolute row and column index in the table, `description` refers to the evidence sentence in the hyperlinked passage, which gives highest similarity score to question, `source` denotes where the entry comes from (e.g. equal/argmax/passage/etc), `score` denotes the score of linked score normalized to [0, 1].

Reasoning This phase is aimed at modeling the multi-hop reasoning between the table and passages, we specifically break down the whole process into three stages, namely the ranking stage $p_f(c|q, \mathcal{C})$, hoping stage $p_h(c'|q, c)$, and the reading comprehension stage

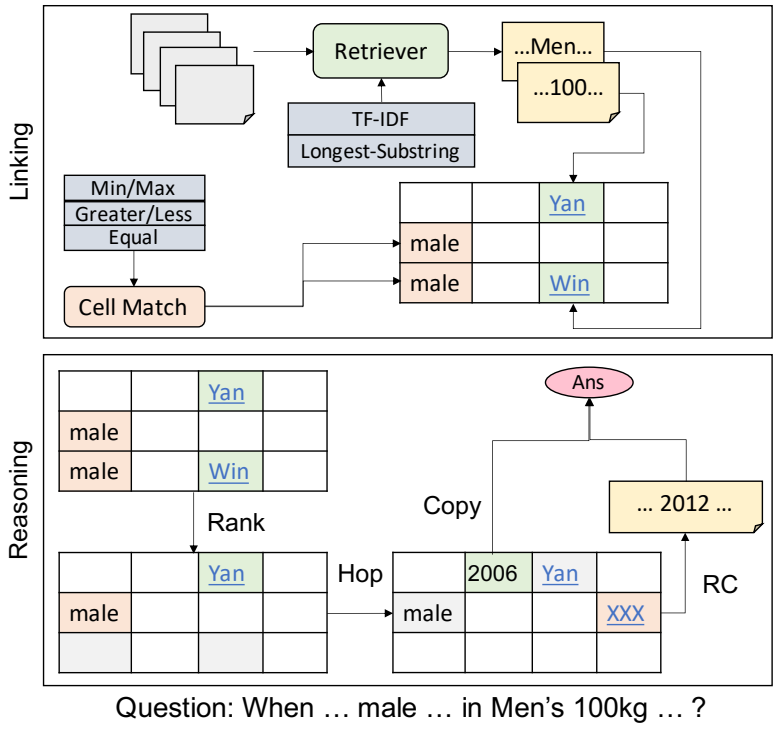

Figure 2.6: Illustration of the proposed model to perform multi-hop reasoning over table and passage.

$p_r(a|P, q)$. These three stages are modeled with three different neural networks. We first design a cell encoding scheme to encode each cell in the table as depicted in Figure 2.5: 1) for "retrieved cells", it contains information for retrieval source and score, 2) for "plain cells" (not retrieved), we set the information in source and score to empty. We concatenate them with their table field and question, and then fed into a encoder module (BERT) to obtain its vector representation H_c.

Ranking model As the "retriever cells" contain many noises, we leverage a ranker model to predict the "correct" linked cells for the next stage. Specifically, this model takes each cell c along with its neighboring $\mathcal{N}_c$ (cells in the same row) and feed them all into the cell encoder to obtain their representations $\{H_c\}$. The representations are aggregated and further fed to a feed-forward neural network to obtain a score s_c, which

is normalized over the whole set of linked cell $\mathcal{C}$ as follows:

$$p_f(c|q,\mathcal{C}) = \frac{exp(s_c)}{\sum_{c' \in \mathcal{C}} exp(s_{c'})} \tag{2.1}$$

Hop model this model takes the predicted cell from the previous stage and then decide which neighboring cell or itself to hop to. Specifically, we represent each hop pair $(c \rightarrow c')$ using their concatenated representation $H_{c,c'} = [H_c, H_{c'}]$. The representation is fed to a feed-forward neural network to obtain a hop score $s_{c,c'}$, which is normalized over all the possible end cells as follows:

$$p_f(c'|q,c) = \frac{exp(s_{c,c'})}{\sum_{c'' \in \mathcal{N}_c \cup c} exp(s_{c,c''})} \tag{2.2}$$

RC model this model finally takes the hopped cell c from last stage and find answer from it. If the cell is not hyperlinked, the RC model will simply output its plain text as the answer, otherwise, the plain text of the cell is prepended to the linked passage $P(c)$ for reading comprehension. The prepended passage P and the question are given as the input to the question answering model to predict the score of answer's start and end index as $g_s(P,q,index)$ and $g_e(P,q,index)$, which are normalized over the whole passage $|P|$ to calculate the likelihood $p_r(a|P,q)$ as follows:

$$p_r(a|P,q) = \frac{exp(g_s(P,q,a_s))}{\sum_{i \in |P|} exp(g_s(P,q,i))} \frac{g_s(P,q,a_e)}{\sum_{i \in |P|} g_e(P,q,i)}$$

where a_s is the start index of answer a and a_e is the end index of answer a.

By breaking the reasoning process into three stages, we manage to cover the Type-I/II/III/VI questions well. For example, the Type-III question first uses the ranking model to select the most likely cell from retrievers, and then use the hop model to jump to neighboring hyperlinked cell, finally use the RC model to extract the answer.

Training & Inference The three-stage decomposition breaks the question answering likelihood $p(a|q,T)$ into the following marginal probability:

$$\sum_{c\in\mathcal{C}} p_f(c|q,\mathcal{C}) \sum_{c'\in\mathcal{N}_c;a\in P(c')} p_f(c'|c,q)p_r(a|P(c'),q)$$

where the marginalization is over all the linked cells c, and all the neighboring cell with answer a in its plain text or linked passages. However, directly maximizing the marginal likelihood is unnecessarily complicated as the marginalization leads to huge computation cost. Therefore, we propose to train the three models independently and then combine them to do inference.

By using the source location of answers, we are able to 1) infer which cells c in the retrieved set $\mathcal{C}$ are valid, which can be applied to train the ranking model, 2) infer which cell it hops to get the answer, which we can be applied to train the hop model. Though the synthesized reasoning paths are somewhat noisy, it is still enough to be used for training the separate models in a weakly supervised manner. For the RC model, we use the passages containing the ground-truth answer to train it. The independent training avoids the marginalization computation to greatly decrease the computation and time cost. During inference, we apply these three models sequentially to get the answer. Specifically, we use greedy search at first two steps to remain only the highest probably cell and finally extract the answer using the RC model.

2.5 Experiments

2.5.1 Experimental Setting

In the linking phase, we set the retrieval threshold τ to a specific value. All the passages having distance lower than τ will be retrieved and fed as input to the reasoning

Model	Dev			Test		
	Table	Passage	Total	Table	Passage	Total
	EM/F1	EM/F1	EM/F1	EM/F1	EM/F1	EM/F1
Table-Only	14.7/19.1	2.4/4.5	8.4/12.1	14.2/18.8	2.6/4.7	8.3/11.7
Passage-Only	9.2/13.5	26.1/32.4	19.5/25.1	8.9/13.8	25.5/32.0	19.1/25.0
Ours-b τ=0.7	51.2/58.6	39.6/46.4	42.9/50.0	50.9/58.6	37.4/45.7	41.8/49.5
Ours-b, τ=0.8	51.3/58.4	40.1/47.6	43.5/50.6	51.7/59.1	37.8/46.0	42.2/49.9
Ours-b, τ=0.9	51.5/58.6	40.5/47.9	43.7/50.9	52.1/59.3	38.1/46.3	42.5/50.2
Ours-l, τ=0.8	54.3/61.4	39.1/45.7	**44.0/50.7**	56.2/63.3	37.5/44.4	**43.8/50.6**

Table 2.5: Experimental results of different models, In-Table refers to the subset of questions which have their answers in the table, In-Passage refers to the subset of questions which have their answer in a certain passage. Ours means HYBRIDER and -b means using BERT-based and -l means using BERT-large.

phase. If there is no passage that has been found with a distance lower than τ, we will simply use the document with the lowest distance as the retrieval result. Increasing τ can increase the recall of correct passages, but also increase the difficulty of the filter model in the reasoning step.

In the reasoning phase, we mainly utilize BERT [39] as our encoder for the cells and passages due to its strong semantic understanding. Specifically, we use four BERT variants provided by Huggingface library[1], namely base-uncased, based-cased, large-uncased, and large-cased. We train the modules all for 3.0 epochs and save their checkpoint file at the end of each epoch. The filtering, hop, and RC models use AdamW [41] optimizer with learning rates of 2e-6, 5e-6, and 3e-5. We held out a small development set for model selection on the saved checkpoints and use the most performant ones in inference.

2.5.2 Evaluation

Following previous work [1], we use exact match (EM) and F1 as two evaluation metrics. F1 metric measures the average overlap between the prediction and ground-truth answers. We assess human performance on a held-out set from the test set contain-

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

ing 500 instances. To evaluate human performance, we distribute each question along with its table to crowd-workers and compare their answer with the ground-truth answer. We obtain an estimated accuracy of EM=88.2 and F1=93.5, which is higher than both SQuAD [1] and HotpotQA [32]. The higher accuracy is due to the In-Table questions (over 40%), which have much lesser ambiguity than the text-span questions.

2.5.3 Experimental Results

We demonstrate the experimental results for different models in Table 2.5, where we list fine-grained accuracy over the questions with answers in the cell and passage separately. The In-Table questions are remarkably simpler than In-Passage question because they do not the RC reasoning step; the overall accuracy is roughly 8-10% higher than its counterpart. With the experimented model variants, the best accuracy is achieved with BERT-large-uncased as backend, which can beat the BERT-base-uncased by roughly 2%. However, its performance is still far lagged behind human performance, leaving ample room for future research.

Heterogeneous Reasoning From Table 2.5, we can clearly observe that using either Table-Only or Passage-Only model achieves a poor accuracy below 20%. In contrast, the proposed HYBRIDER can achieve up to 50% EM increase by leveraging both structured and unstructured data during reasoning. It strongly supports the necessity to do heterogeneous reasoning in HybridQA.

Retriever Threshold We also experiment with a different τ threshold. Having an aggressive retriever increases the recall of the mentioned cells, but it increases the burden for the ranking model. Having a passive retriever can guarantee the precision of predicted cells, but it also potentially miss evidence for the following reasoning phase. There exist

trade-offs between these different modes. In Table 2.5, we experiment with different τ during the retrieval stage and find that the model is rather stable, which means the model is quite insensitive regarding different threshold values.

2.5.4 Error Analysis

To analyze the cause of the errors in HYBRIDER, we propose to break down into four types as Figure 2.7. Concretely, linking error is caused by the retriever/linker, which fails to retrieve the most relevant cell in the linking phase. In the reasoning phase: 1) ranking error is caused by the ranking model, which fails to assign a high score to the correct retrieved cell. 2) hop error occurs when the correctly ranked cell couldn't hop to the answer cell. 3) RC error refers to the hoped cell is correct, but the RC model fails to extract the correct text span from it. We perform our analysis on the full dev set based

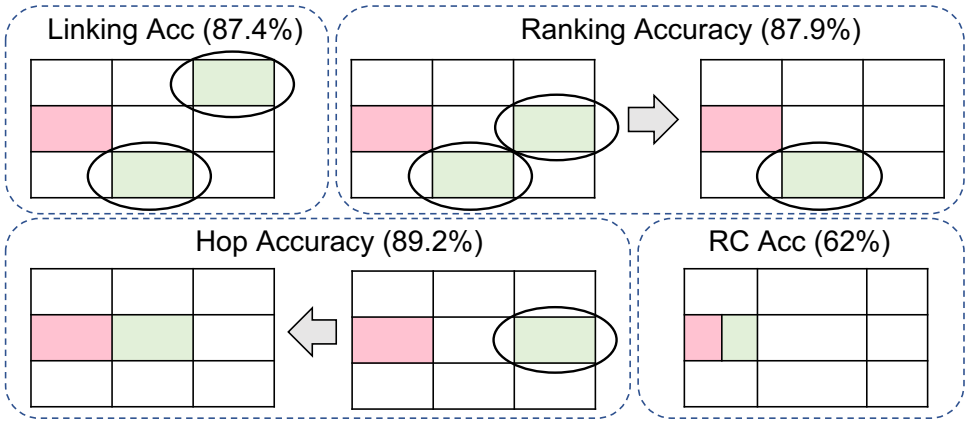

Figure 2.7: The error of HYBRIDER is based on its stages. Pink cell means the answer cell; green means the model's prediction; circle means the current cell.

on the bert-large-uncased model (τ=0.8), as indicated in Figure 2.7, the errors are quite uniformly distributed into the four categories except the reading comprehension step is slightly more erroneous. Based on the step-wise error, we can compute its product as $87.4\% \times 87.9\% \times 89.2\% \times 61.9\% \approx 42\%$ and find that the result consistent well the overall accuracy, which demonstrates the necessity to perform each reasoning step correctly. Such

error cascading makes the problem extremely difficult than the previous homogeneous question answering problems.

By breaking down the reasoning into steps, HYBRIDER layouts strong explainability about its rationale, but it also causes error propagation, i.e., the mistakes made in the earlier stage are non-reversible in the following stage. We believe future research on building an end-to-end reasoning model could alleviate such an error propagation problem between different stages in HYBRIDER.

2.6 Related Work

Text-Based QA Since the surge of SQuAD [1] dataset, there have been numerous efforts to tackle the machine reading comprehension problem. Different datasets like DrQA [40], TriviaQA [19], SearchQA [42] and DROP [31] have been proposed. As the SQuAD [1] questions that are relatively simple because they usually require no more than one sentence in the paragraph to answer. The following datasets further challenge the QA model's capability to handle different scenarios like open-domain, long context, multi-hop, discrete operations, etc. There has been a huge success in proving that the deep learning model can show strong competence in terms of understanding text-only evidence. Unlike these datasets, HYBRIDQA leverages structured information in the evidence form, where the existing models are not able to handle, which distinguishes it from the other datasets.

KB/Table-Based QA Structured knowledge is known as unambiguous and compositional, which absorbs lots of attention to the QA system built on KB/Tables. There have been multiple datasets like WebQuestion [20], ComplexWebQuestions [23], WebQuestionSP [21] on using FreeBase to answer natural questions. Besides KB, structured or

semi-structured tables are also an interesting form. Different datasets like WikiTableQuestions [24], WikiSQL [25], SPIDER [26], TabFact [11] have been proposed to build models which can interact with such structured information. However, both KB and tables are known to suffer from low coverage issues. Therefore, HybridQA combine tables with text as complementary information to answer natural questions.

Hybrid QA There are some pioneering studies on designing hybrid question answering systems to aggregate heterogeneous information. GRAFT [28] proposes to use the early fusion system and use heuristics to build a question-specific subgraph that contains sentences from corpus and entities, facts from KB. PullNet [27] improves over GRAFT to use an integrated framework that dynamically learns to retrieve and reason over heterogeneous information to find the best answers. More recently, KAReader [29] proposes to reformulate the questions in latent space by reading retrieved text snippets under KB-incomplete cases. These models simulate a 'fake' KB-incomplete scenario by masking out triples from KB. In contrast, the questions in HybridQA are inherently hybrid in the sense that it requires both information forms to reason, which makes our testbed more realistic than the other settings.

Chapter 3

Open Question Answering over Hybrid Web Knowledge

3.1 Introduction

In the previous chapter, we have discussed how to perform reasoning over the given heterogeneous evidence data. Our proposed method can already achieve decent performance to integrate structured and unstructured forms, which indicates the potential of hybrid reasoning method in real-world applications. However, the previous chapter assumes that the heterogeneous evidence for question answering is already known a priori. Such an assumption is not realistic in building real-world interfaces like Apple Siri, Amazon Alexa, etc, where we do not have access to the ground-truth evidence and have to search over the Web for find such supportive evidence in various forms.

In order to deal with such a more realistic setting, the standard approach Figure 3.1 for question answering is to first scan the Web to retrieve the most relevant heterogeneous evidence and then perform reasoning on top of it. More recently, such a pipeline approach is known as open-domain question answering [40, 19]. Practically, we use the Wikipedia

as the main Web resource due to its well-defined structure, but the approaches can easily generalize to broader domains like Forms, News, etc. In this chapter, we consider such open-domain answering question over heterogeneous knowledge from the Web.

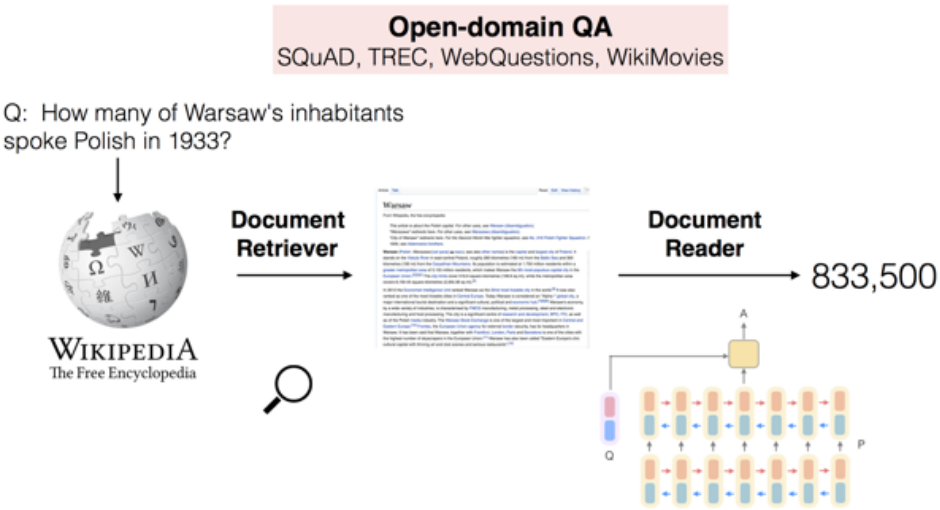

Figure 3.1: Open-Domain Question Answering Paradigm.

For this purpose, we construct a new dataset as our benchmark. Open Table-and-Text Question Answering (OTT-QA). OTT-QA is built on top of the HybridQA dataset [8], and like HybridQA, OTT-QA questions are multi-hop questions which require aggregating information from both tables and text to answer. However, unlike HybridQA, OTT-QA requires the system to *retrieve* relevant tables and text — in contrast, in HybridQA, the ground truth tables and textual passages required for each question are given. To produce OTT-QA's questions, we begin by re-annotating the questions from HybridQA to 'decontextualize' them—i.e., we make questions suitable for the open-domain setting so that unique answers can be determined from the question alone, without needing context from the provided text and tables. We then add new questions to remove potential biases. After these steps, OTT-QA contains $45K$ human-annotated questions that require retrieving and aggregating information over tables and text from the whole

Wikipedia. Examples from OTT-QA are depicted in Figure 3.2. Note the table and passages contain non-overlapping information, and both of them must be properly understood to answer the question. For example, the question has a low lexical overlap with the passage about the `'Lakers'`, and it needs the table as the bridge to retrieve this passage. Such cross-modality multi-hop retrieval features OTT-QA.

OTT-QA is distinguished from the existing QA datasets in two aspects. Existing table-based QA datasets [24, 26, 8] operates in the closed setting without requiring any retrieval, whereas most existing open QA datasets [19, 32] require only text retrieval, not table retrieval. One dataset, Natural Questions (NQ) [33] includes some tabular information in its corpus, but the tables are nearly always of a restricted type (infobox tables with only a single row). In contrast, OTT-QA models require retrieving *both* tabular data and text, and unlike the NQ dataset, requires information fusion from text and tables in non-trivial ways. OTT-QA poses novel and realistic challenges to both the retriever and reader in open QA though the questions are less natural than the real queries from NQ [33]. Retrievers for OTT-QA need to consider two information formats, making the search space larger. Even worse, as questions in OTT-QA often require multi-hop inference, one round of retrieval is often not enough. Readers for OTT-QA also need to aggregate a significant amount of knowledge-intensive information, compared to other reader models: a single table in OTT-QA has an average length of over 300 words. Moreover, readers are often expected to process multiple retrieved units due to the uncertainty in retrieval, which makes it difficult to design strong reader models [39, 43] with a length limit of 512 tokens.

The baseline system uses an iterative retriever [44, 45, 46, 47, 48] and a BERT reader [39]. The iterative retriever explores multiple evidence documents iteratively, interacting with the candidate pool to gradually reformulate the query. Beam search is used to find multiple subsets of documents that may contain all the required evidence,

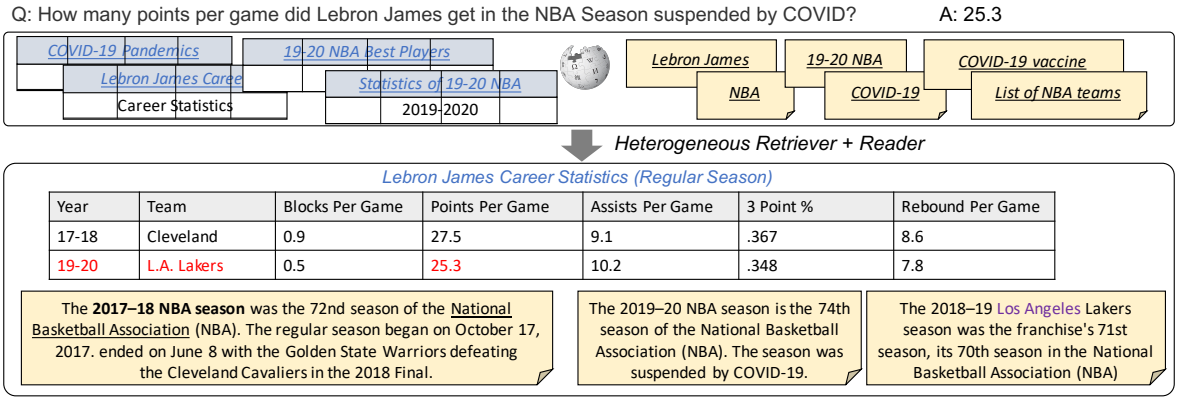

Figure 3.2: The problem setting: A OTT-QA model needs to retrieve from two candidate pools and then perform multi-hop reasoning to find answers.

and each subset is then fed to the BERT reader to predict the answer span. The highest-scored prediction is chosen as the answer. The iterative retriever needs to re-encode the query with a big transformer and re-search over the candidate pool, such a procedure (especially dense) can be computationally expensive. Furthermore, the BERT reader fails to capture a global overview of the retrieved documents, which leads to bad local optimum in the model prediction.

We propose a more sophisticated system that addresses these challenges with two novel strategies: namely *fusion* retrieval and *cross-block* reading. The **fusion retriever** first pre-aligns the table segments to their highly related passages, using pre-trained entity linking system. Then, the aligned table segments and passages are grouped as a *fused block*, which contains aggregated information from two modalities; hence, compared to the previous documents, it contains richer context to benefit the following retrieval. We view the fused block as the basic unit to be retrieved, and instead of performing multiple runs of retrieval iteratively, the fusion retriever is used once to retrieve the top K fused blocks; however, due to errors in fusion and retrieval, the retrieved top-1 fused block might not contain the necessary information to answer the given question. We thus also propose a **cross-block reader** based on a sparse-attention based transformer

architecture [49, 50], which can process extremely long sequences efficiently. We use the cross-block reader to read all the top-K retrieved fused blocks jointly. Both strategies have proven effective compared to the baseline system: the best model combining the two strategies improves the accuracy of the baseline system by a large margin.

3.2 Background

The aim of an open QA system is to extract an answer to a question q from a given large corpus. Most open QA models are retriever-reader models, which extract answers in two steps: retrieval and reading. In the *retrieval* step, a retrieval model f is used to retrieve a set of passages from the text corpus. In the *reading* step, the reader is then used to extract the answer from them.

Retrieval Function There are two commonly-used types of retrieval function f: sparse retrievers and dense retrievers. Our sparse retriever uses a unigram-based BM-25 score to retrieve an evidence unit b from the candidate pool $\mathbb{B}$. Our dense retrieval function is a dual-encoder model [51], and we follow [52, 53] for the dual encoder design. The query and the passage are encoded with separate Transformers. As in [39], the vector corresponding to the first token, `[CLS]`, is used as a "pooled" representation of the sequence. The dense retrieval function is the dot product between $h_q = \texttt{BERT}_\texttt{Q}(q)$[CLS] and $h_b = \texttt{BERT}_\texttt{B}(b)$[CLS] for each evidence block b in the candidate corpus—i.e., the scoring function is $f(q, b) = h_q^T h_b$, which can viewed as finding the nearest neighbor in vector space. In the multi-hop open QA setting [32], an iterative retrieval function [44, 46, 47] is proposed, which defines the retrieval process as an auto-regressive formula. Our iterative retriever function is denoted as $f([q, b_1, \cdots, b_{j-1}], b_j)$, which appends the previous $j-1$ rounds of retrieval to the original q in in the j-th round of retrieval. Beam search is used in test time.

Single-Block Reader Due to the uncertainty in retrieval, the top-1 document might not always contain the answer. Existing models normally retrieve the top-k documents and feed them to the reader for span selection. The standard reader [40, 19] aims to extract a span from each of the retrieved blocks b_i and assign a confidence $f(q, b_i)f_{read}(a|q, b_i)$ to it, with $f(q, b_i)$ indicating the retrieval probability and $f_{read}(a|q, b_i)$ denoting the span selection probability by reader. Multiple answers $\{a_1, \cdots, a_k\}$ are ranked with this confidence score and the highest scored answer span $\hat{a}$ is the final answer. Note that the reader needs to run k times, once for each of the top-k retrievals. We refer to this model as the *single-block reader* and use it as our baseline.

HybridQA HybridQA [8], a closed-domain QA dataset, is the most related to ours. During the annotation of HybridQA, a table T and its relevant passages $\{P_1, \cdots, P_N\}$ (surrounding text and hyperlinked passage) are given to a crowd worker to write questions which necessarily require both the passage and table information to answer. The original dataset contains 72K multi-hop questions paired with 13K tables with their paired passages. During training/testing time, the ground-truth tables and passages are given to a model, HYBRIDER, to find the final answer. HYBRIDER also serves as an important baseline for the following sections.

3.3 Task and Dataset

In OTT-QA, the retrieval corpus consists of a set of table candidates $\mathbb{B}_T$ and a set of passage candidates $\mathbb{B}_P$. The task is to answer question q by extracting answer strings from blocks $b \in \mathbb{B}_T \cup \mathbb{B}_P$, where b can be either textual and tabular data. We adopt the standard exact match (EM) and F1 scores [32] for evaluation. Different from HybridQA, OTT-QA's table candidates are web tables *without* hyperlinks provided. This decision

was made to make the problem setting more general, as otherwise systems that solve OTT-QA could only be applied to high-quality data in Wikipedia. However, in OTT-QA, we provide hyperlinks in the training subset, but not dev/test set. Removing hyperlinks in tables makes the overall task much more challenging, but makes the final systems applicable to more general domains. Thus, an OTT-QA model needs to jointly retrieve both tables and text, without abusing gold hyperlinks, and then aggregate them to find the answer.

Candidate Pool For our table collection $\mathbb{B}_T$, we extracted all Wikipedia regular tables with their metadata including page title, page section title, and section text. The metadata, denoted T_M, is essential for de-contextualization. We obtain a table corpus containing over $400k$ high-quality tables with an average length of 320 words including metadata. For the text passage collection $\mathbb{B}_P$, we crawl English Wikipedia dump pages and filter out noisy pages. We follow HybridQA [8] and only keep a maximum of 12 sentences in the introduction section as the passage. We obtain a corpus containing over 5 million passages, with an average of 94 words.

Notation We define each table as a matrix T, which consists of cells $T_{i,j}$ with i specifying the row, and j specifying the column. Each cell $T_{i,j}$ could be a number, date, phrase or even sentence due to its semi-structured nature. However, a single complete table with structured representation [54] can easily exceed the 512-token limit, which poses great challenges to the downstream reader to process top-K retrieval. Hence we propose to decompose each table T into multiple rows R_i, which are combined with the headers, metadata, and global max/min information from the original table as a **table segment**. The table segment is used as the basic retrieval block. This decomposition procedure increases candidate $\mathbb{B}_T$ from $400k$ to 5 million, making the retrieval problems

even more fine-grained and more challenging. In summary, we build a candidate pool of 5 million table segments $\mathbb{B}_T$ and a pool of 5 million passages $\mathbb{B}_P$. We denote as $\mathbb{B}$ as our full candidate pool, which our model needs to find the block b (a table segment or a passage) containing the answer span.

3.3.1 Question and Answer Annotations

Our question and answer pairs are built upon the existing HybridQA [8] dataset, with several significant changes. First, crowd workers 'decontextualize' the questions so that they are not under-specified or context-dependent, and thus suitable for the open setting. Second, we add more questions to the development/test set to remove possible annotation bias. During annotation, we adopt strict quality control[1].

Decontextualization Most questions in HybridQA are contextualized with a given table and several passages, with corresponding questions written by crowd workers. Often, the crowd-sourced questions assume the context. For example, the questions might contain the words `"the players"` because the given table is about `"Netherlands players"`. We thus needed 'de-contextualize' [55] the original context-dependent questions, so they could serve as standalone questions, specific enough to imply a unique answer relative to the corpus. To discourage excessive unwanted modification, we enforce a two-step annotation procedure, as depicted in Figure 3.3. In the first phase, the worker is only allowed to insert **minimum** words or phrases (or replace pronouns) into the questions based on the information presented by Wikipedia Title, Section Title, and Section Text to make the question have a unique answer. After this step, we often potentially obtain

[1]The collection is conducted on an established crowdsourcing platform with annotators from countries with English as the native language. The annotators were required to meet the requirements: 1) a native speaker in an English-speaking country 2) having an approval rate of over 95% and 3) having at least 500 approved jobs.

overly-complicated questions that are artificial and unnatural. Therefore, we manually selected the worst 25% questions and sent them back to make them more concise and natural.

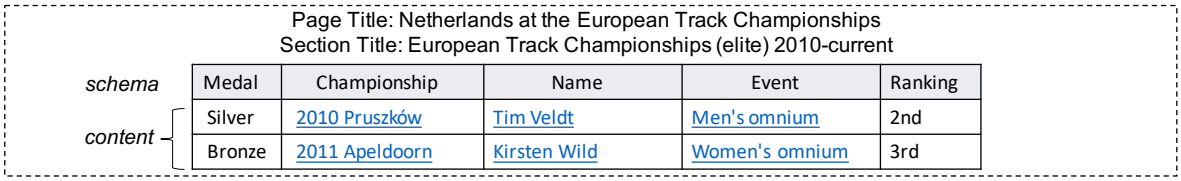

Figure 3.3: The 'de-contextualization' annotation phase of OTT-QA. In the first step, the annotator is restricted to add phrases from the context. In the second step, the annotator is specifically requested to make the sentence more concise and natural.

Additional Evaluation Examples As all the questions from HybridQA are based on the 13k tables from the HybridQA set, no questions are asked about the newly crawled 400k tables. This potentially generates unwanted statistical biases or artifacts for the model to exploit, and potentially biases the final evaluation results. Therefore, we randomly sampled another 1100 tables from the newly crawled tables, and follow the original annotation process used by HybridQA to re-collect 2200 new questions. These new questions were mainly used in the dev/test set. Below we refer to the subset of tables used by original HybridQA as the *in-domain* tables.

Distant Supervision Signals For the in-domain tables ($\approx 8k$), the cell-wise hyperlinks are provided in OTT-QA as a potential signal for supervision. We use $H_{i,j} = \{b_1, b_2, \ldots \in \mathbb{B}_P\}$ to denote the hyperlinks in cell $T_{i,j}$. Since in HybridQA the oracle fine-grained answer span is not explicitly annotated, we approximate this by traversing the table and hyperlinked pasasages to find all exact matches. This process contains some

noise—a manual study reveals that it roughly contains 15% error. We use this 'weakly-supervised' fine-grained information to train our models. We denote the 'approximate' block of the answer span for answer a as b_a, and use it to train our model.

3.3.2 Dataset Statistics

After annotation, we sampled roughly 2K questions from the in-domain HybridQA dataset, and then mix them with the newly collected out-domain questions to construct our dev and test sets. Finally, we have 41,469 questions in the training set, 2,214 questions in the dev set, and 2,158 questions in the test set. We conduct more in-detailed analysis over the reasoning types and show that a remarkable difference from original HybridQA is that a proportion of questions actually have multiple plausible inference chains in the open-domain setting.

3.4 Model

Our model for OTT-QA is a retriever-reader model with new designs for both retriever and reader. As discussed briefly above, we propose to use a fusion retriever instead of using a standard iterative retrieve, and we also propose to use cross-block readers to replace a standard single-block reader.

3.4.1 Fusion Retriever

Iterative retrieval (Figure 3.4, Left) has the following issues. First, iterative retrieval training often requires having supervision signals for every retrieval step to reach good performance, which is not available in OTT-QA. The iterative retrieval also suffers from the problem of error propagation, as early mistakes can propagate to later retrieval stages.

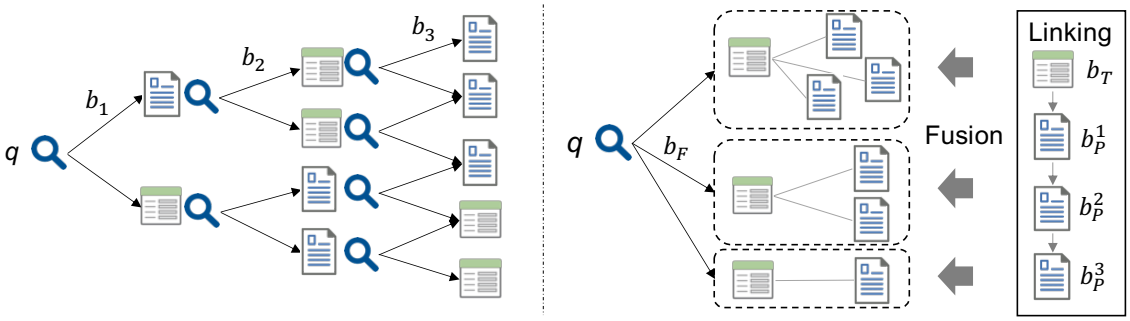

Figure 3.4: **Left:** Iterative 3-step retrieval over individual blocks (baseline). **Right:** Fusion 1-step retrieval over fused groups, which greatly lowers the cost of iterative encoding and retrieving.

Finally, the computation cost for applying a dual-encoder for iterative retrieval is very high, as for every stage, the query embedding has to be re-encoded to include the entire retrieval history.

We propose an alternative strategy to replace multi-step retrieval, namely fusion retrieval (Figure 3.4, Right). In the fusion retriever, we first use an 'early fusion' strategy to group relevant heterogeneous data before retrieval. The fusion procedure groups several highly-relevant blocks from different modalities as a self-contained group (fused block), which provides more clues for the retriever to utilize. Early fusion is very important for retrieving table segments, which often have incomplete context by themselves. The early fusion process aims to fuse a table segment and relevant passages into a group. Here we propose to fuse entities mentioned in a table segment to the appropriate passages for those entities; this is similar to document expansion based on a traditional entity linking step. The problem is challenging due to the mismatch between the lexical forms from the table (which for brevity are often abbreviated) and the relevant passage titles. For example, a cell in the table of `"NCAA Division I Men's Football Tournament"` contains the term `"Penn State"`. Directly matching `"Penn State"` against the passage corpus will lead to `"Penn State University"` rather than the ground-truth hyperlinked entity, named `"Penn State Nittany Lions football"`. Therefore, we propose an additional augmenta-

tion step, which takes in a table segment block b_T and generates a sequence of augmented queries $q_1, q_2, \cdots, q_n$ token by token to make the queries more similar to the passage title. The augmented queries are then used to search for nearest neighbors in the passage corpus $\mathbb{B}_P$ using BM25 as the final entity linking step. The query augmentation is implemented with a GPT-2 model [56], fine-tuned on the supervised pairs of (table segment, hyperlink) from the in-domain tables. Each b_T is fed to find its companions $b_P^1, \cdots, b_P^n$, they are collectively called b_F.

We follow the standard dual-encoder setting (section 3.2) and the only difference is that we replace the input of the block encoder with $h_b = \texttt{BERT}_\texttt{B}([b_T, b_P^1, \cdots, b_P^n])$., which captures the cross-attention between the table and the text within a block. The fused embedding contains richer context from both modalities to complement each other. The retriever only needs to retrieve once from the candidate pool, which dramatically decreases the complexity compared to the existing iterative retrievers.

To enhance the neural retrieval system to retrieve fused blocks, we apply the Inverse Cloze Task (ICT) [52] pretraining task on the corpus of fused blocks. ICT is a way to generate pseudo-training data for dense retrieval. Unlike standard document-wise ICT, our fused block contains both table segments and multiple passages. Given a fused block b_F, we generate the pseudo-query in the following way: 1) we first corrupt the table segment by randomly dropping half of the words from the table metadata and cells to obtain a partial table segment $\hat{b}_T$. 2) We then randomly sample a sentence $\hat{b}_P$ from the fused passage. We combine $\hat{b}_T$ and $\hat{b}_P$ as a pseudo query $\hat{q}$ and pair it with the original fused block b_F as pre-training data. The pre-training data is applied to enhance the dual encoder's ability to select lexically matched documents. After pre-training, the retriever is fine-tuned on OTT-QA. Finally, at inference time, the retriever is used to retrieve the top K fused blocks for a question, which are then passed to the reading comprehension model for extractive answer prediction.

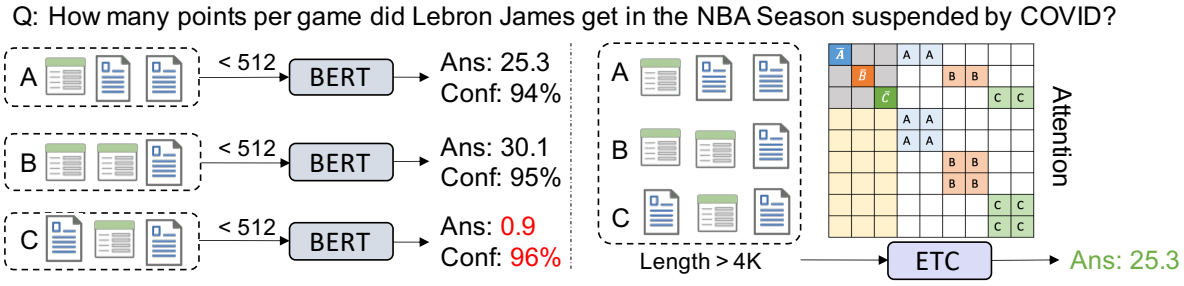

Figure 3.5: **Left:** Single-block reader with input shorter than 512 tokens (baseline). **Right:** Cross-block reader with length over 4K tokens, and $\bar{A}$ denotes the global state assigned to local block A. The single-block reader is stuck at local optimum, while cross-block reader outputs global optimum.

3.4.2 Cross-Block Reader

The reader typically needs to process the top-k retrieved blocks returned by the retriever to extract the best answer, as the top-1 block might not contain enough evidence to answer the question. As demonstrated in Figure 3.5, the cross-block reader aims to address this issue by using cross attention between different blocks to model their dependencies. To obtain the cross-block reader, we take the pre-trained long-range sparse attention transformer (ETC) [49], which can accept up to 4096 tokens as input, and then fine-tune the model on the distant supervision data. During training, the ground truth (fused) blocks are mixed with hard negative blocks from the retriever. We take the top-k retrieval results to fill the 4096 token space (roughly 15 fused blocks).

Cross-attention between blocks allows a much more powerful way to aggregate information across the k retrieved blocks compared to the single-block reader, especially when the blocks are fused. This is feasible because of the design of the sparse attention structure in ETC, which can constrain the attention of each token to its neighboring tokens within a local radius in its local block. Such sparse attention can decrease the attention computation complexity from quadratic $\mathcal{O}(N^2)$ to linear $\mathcal{O}(N|R|)$, where $|R|$ is

the local radius (where $N = 4096$ and $|R| = 84$ in our experiments). To allow cross-block interaction, ETC assigns a global state for each local block in the long sequence, and blocks can attention to each other through multiple layers of such global-local structures.

3.5 Experiments

All of our code is based on Tensorflow [57]. For the retriever part, the sparse retriever is built on top of DrQA [40] with unigram features, and the dense retriever is built with BERT. The single-block retriever is based on BERT-uncased, and the cross-block reader is based on ETC [49]. Both of them consist of 12 layers with a hidden size of 768, the minor differences in the relative positional embedding used in ETC. All the models are trained with a learning rate of 1e-5 optimized by AdamW [41]. We use in-batch negatives [52] to train our dense retrievers. In fusion retriever, we use the 'fused' block containing the 'approximate' answer block b_a as the positive instance. In iterative retriever, since the auto-regressive model $f(b_j|q, b_1, \cdots, b_{j-1})$ requires fine-grained inference chain for step-wise supervision, which is not given in OTT-QA. We apply lexical match based heuristics to synthesize inference chains as weakly supervised training data. For all the dense retrievers, we pre-train with 10K steps using the generated pseudo query and then fine-tune them another 10K step using a batch size of 2048. For the cross-block reader, we fine-tune with a batch size of 64. Both are using 16 cloud TPUs.

Main Results In our experiments, we experiment with different types of retriever and reader models under both sparse and dense setting, the details are described as follows:

- HYBRIDER: this model, designed for closed domain HybridQA questions, is one baseline. Since this model requires a ground truth table with its hyperlinks to do modularized reasoning, we use BM25 to retrieve the most relevant table and

Retriever	Dev-Sparse		Dev-Dense		Test-Best	
Model	EM	F1	EM	F1	EM	F1
HYBRIDER (Top-1) [8]	8.7	10.9	8.9	11.3	8.4	10.6
HYBRIDER (best Top-K) [8]	9.9	12.2	10.3	13.0	9.7	12.8
Iter-Retrieval + Single-Block Reader	9.8	13.3	7.9	11.1	9.6	13.1
Fusion-Retrieval + Single-Block Reader	14.3	17.8	13.8	17.2	13.4	16.9
Iter-Retrieval + Cross-Block Reader	17.1	20.7	14.4	18.5	16.9	20.9
Fusion-Retrieval + Cross-Block Reader	27.7	31.8	**28.1**	**32.5**	**27.2**	**31.5**
Ablations	EM	F1	EM	F1	EM	F1
Table-only Retrieval + Cross-Block Reader	4.6	6.9	4.9	7.2	4.4	7.0
Text-only Retrieval + Cross-Block Reader	8.2	12.4	8.9	12.8	8.8	12.1
Groundtruth Table/Text + HYBRIDER	44.1	50.8	44.1	50.8	43.0	49.8

Table 3.1: **Main Results**. We conduct experiments with both sparse and dense retrievers using the dev set, and then select the best setting to report the test set results (as indicated by the word "Best"). Fusion-Retriever and Cross-Block Reader are combined to obtain the highest score.

passages to reconstruct an 'approximated' input for this model. We experiment with top-1,2,3,4 cases where we use the answer with the highest confidence as the final result. We also directly feed the ground-truth table and hyperlinks to HYBRIDER, which roughly estimates an upper limit of this task.

- Iterative-Retriever (Sparse): We use a 2-step iterative retriever: in the first step, we apply the question to retrieve the top-10 table segments and top-10 passages. In the second step, we use each retrieved table segment to retrieve its related top-5 passages and concatenate each retrieved passage title with the original question to retrieve the top-5 table segments. We merge and calculate the retrieval score of each unique block and rank them by their score. For the single-block reader, we split the retrieved blocks into 512-token chunks and feed them to the BERT reader. For the cross-block reader, we truncate the top 4096 subword tokens and only feed these tokens to reader.

- Iterative-Retriever (Dense): We use a 3-step iterative retriever. In the first step, we encode the question and retrieve the top-8 blocks (either table segment or passage); in the second step, we concatenate the previous retrieved block and the question to re-encode the query vector to further retrieve top-4 blocks; similarly, the last step retrieves top-2 blocks.

- Fusion Retriever (Iterative): We use a sparse retriever to directly retrieve the top-15 fused blocks based on bag-of-words BM25 score, and then split it into individual table segments and passage blocks. Since passage could be associated with multiple fused blocks, we merge duplicate blocks and use their summed score. Finally, we rank each block based on its merged retrieval score and truncate the first 4096 subword tokens for the next step.

- Fusion Retriever (Iterative): We use a dual-encoder dense retriever to directly retrieve the top-15 fused blocks, and then follow the same procedure as above. Without specifying the dense retriever uses ICT for pre-training by default.

- Fusion Retriever w/o ICT and w/o GPT-2: these two ablation studies are aimed to show the effectiveness of our proposed ICT pre-training and query augmentation.

The main results are presented in Table 3.1. First, we can observe that best HYBRIDER top-2 can only achieve a comprised exact match of 9.9% while the oracle HYBRIDER can obtain a score of 44%, which reflects the difficulties of the hybrid retrieval in our dataset. We restrain the retriever to only retrieve table and text to answer the questions and report their results in Table 3.1, even with the strong cross-block reader, the model only obtains 10% EM. These experiments demonstrate the necessity to integrate information from both forms in OTT-QA.

By combining the standard iterative retriever and single-block reader, we can slightly

improve the score can to roughly 10%. By replacing the iterative retriever with the proposed sparse fusion retriever, the EM score can reach 14%, a 4.5% absolute improvement. By replacing the single-block with the proposed cross-block reader, the EM score can reach 17% , a 7% absolute improvement. However, by combining the two strategies, the final EM score can reach 28%, with an 18% absolute improvement, which is greater than the sum of individual improvements. The observation suggests the two components can affect each other in a positive way. We conjecture that the fusion retriever is more likely to retrieve mutually-supportive blocks in a group, which makes the multi-hop reasoning across different blocks easier for the following cross-block reader. In comparison, the iterative retriever retrieves isolated table segments and passages separately, which can easily miss out on the bridging evidence for building the complete reasoning chain. Thus, the cross-block reader cannot maximize its advantage in reasoning across blocks.

By removing the ICT pre-training and query augmentation, we observe that the Dev-EM score drops to 24.6%. By removing the GPT-2 query augmentation, the Dev-EM performance drops to 22.1%. These two results indicate the effectiveness of the proposed two strategies. By replacing the predicted hyperlinks with the oracle links, the fusion model performance can increase by 7% EM. This indicates that there is still plenty of room to improve for the table-passage fusion model.

Linker/Retriever Results To understand the results more, we evaluate the standalone table-passage entity linking accuracy and retriever recall.

We consider the following linking models: a) BM25 model, which directly uses the cell value to retrieve passages based on their titles without query augmentation, b) a Dual-Encoder model, which encodes the cell value and meta information into a query vector to compute dot-product over all the passage candidate to retrieve, c) a GPT-2 model, which first augments the cell value by the context and then uses BM25. We demonstrate

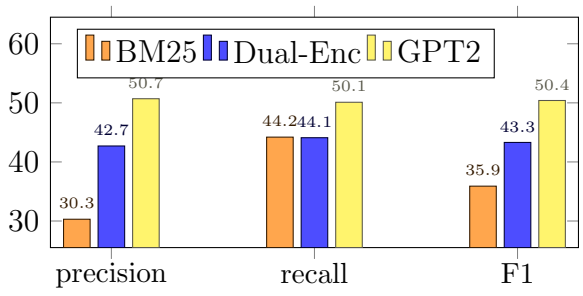

Figure 3.6: Entity linker performance (F1).

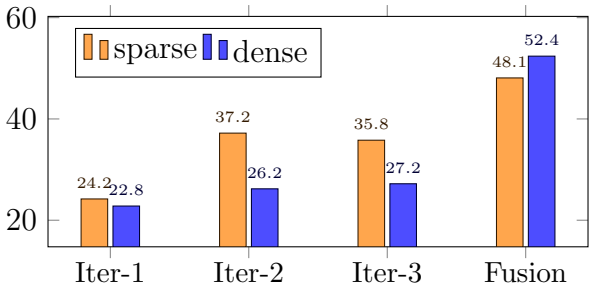

Figure 3.7: Retriever performance (HITS).

our findings in Figure 3.6, and evaluate with table-segment-wise F1 score. We observe that directly using BM25 leads to compromised precision of 30.3%, which is mainly due to the lack of context information. By using a dual-encoder retriever, the precision can be improved to 42%. However, many table segments have either zero or multiple linked passages and can be better modeled by an auto-regressive retrieval process.

We use HITS@4K is used to measure the retriever performance, which indicates the chance of ground truth block existing in the retrieved 4096 subword tokens. The results are reported in Figure 3.7. We vary the steps of iterative retrievers to show the necessity of multi-hop retrieval in OTT-QA. We observe that the 1-step retrieval has the lowest recall because the answer block in OTT-QA normally has a lower lexical overlap with the query. Adding the second retrieval step can greatly improve the recall, but adding the third retrieval hop has very little impact. In contrast to the iterative retriever, the fusion retriever can consistently improve the performance over the iterative setting for both sparse and dense setting. The sparse setting can rise from 35.8% to 48.1% indicating the advantage of 'early' fusion. The dense retriever's improvement is more dramatic (from 27.2% to 52.4%). We believe this is because the iterative retriever heavily relies on noisy synthetic inference chain data, while the fusion retriever does not require such a fine-grained supervision signal, thus less prone to noise.

Model	Dev-EM	Dev-F1
Semantic Retriever [59]	46.5	58.8
Cognitive Graph [47]	37.6	49.4
DocKIT [60]	42.1	51.7
Transformer-XH [58]	50.2	62.4
RNN-Retrieval [48] (BERT base)	52.7	65.8
Ours (Fusion Retriever + Cross-Block Reader)	50.4	61.7

Table 3.2: **HotpotQA Results**. We conduct experiments on dev-set of HotpotQA, and then compare with the state-of-the art models with similar model size.

Generalization Results To further demonstrate our model's effectiveness on more general open-domain question answering, we also test on purely text-based multi-hop question answering dataset HotpotQA [32]. In this dataset, we follow the same procedure to group passage individual blocks as grouped blocks offline. The retrieved blocks are fed to the cross-block readers to allow interaction between retrieved blocks. We demonstrate our results on the dev-set in Table 3.2, where we compare against the other models using similar model size (BERT-base). As can be seen, our model is able to achieve competitive performance with these state-of-the-art models. However, our model is more efficient in a sense that our model does not require any additional re-ranking step over the retrieved blocks. Both Transformer-XH [58] and RNN-Retrieval [48] require expensive re-ranking over a large amount of retrieved reasoning graphs to select the most salient documents for the next stage (reader). In contrast, our model directly feeds the retrieved blocks to reader, which greatly simplifies and accelerates the system.

3.6 Related Work

Table Retrieval Tables are pervasive on the Web, there have been some studies on mining web tables to answer open-domain questions [61, 62]. In [61], the authors have proposed a pipeline framework to first detect the topic entity and then generate a can-

didate chain, finally ranking chains to predict the answer cell. In [62], the authors investigate different similarity matching features to retrieve tables from the web. Our work is significantly different from these two studies in two aspects: 1) the previous works use private small-scale datasets while we collect a large-scale dataset and release it for public use, 2) the previous studies are restricted to only using tables as evidence, while our work considers a more realistic and challenging setting with both table and text corpus. Tables have been a ubiquitous information representation form to express semi-structured information. There has been a long-standing effort to utilize tables in natural language processing applications [24, 25, 26, 55, 63]. However, these existing tasks are restricted to in-domain cases without requiring any retrieval, and our work is the first to investigate retrieving web tables for downstream tasks. Another pair of related works are TAPAS [54] and TABERT [64], which investigate joint pre-training over textual and tabular data. Our method draws inspiration from these models, and also uses special tokens and embeddings to encode spatial and logical operations inside tables.

Long Range Transformer Recently, many transformer variants to resolve the $\mathcal{O}(n^2)$ attention cost have been proposed including Sparse Attention [65], Reformer [66], Routing Transformer [67], Longformer [68] and ETC [49]. These different transformer models apply hierarchical architecture, local-sensitive hashing, global-local state to decrease the attention complexity to nearly linear. Our cross-block reader is based on ETC [49], but unlike prior works that process one long document for QA, our task requires reading multiple blocks containing both structured and unstructured data. To handle the long sequence of retrieved documents in open-domain question answering, Fusion-in-Decoder [69] has been proposed to replace the extractive model with an encoder-decoder generative model. The long sequence of passages are split and encoded independently to decrease the computation complexity, but the decoder still uses full attention over the

tens of thousands of encoded vectors to generate the answer token by token. Such full-attention can decrease the decoding speed by an order of magnitude, while our sparse-attention-based cross-block reader can still maintain the same speed as the standard BERT model.

Chapter 4

Question Answering over Multimodal Web Knowledge

4.1 Introduction

In the previous two chapters, we have discussed how to perform reasoning over structured and unstructured heterogeneous data on the Web. We demonstrate that by utilizing more knowledge forms on the Web, the performance of the existing QA systems can be dramatically improved. To further increase the coverage, can we utilize the massive amount of visual data from the Web? Since visual data is also a pervasive representation on the Web to distribute information, it's beneficial to design models to leverage these visual data to answer questions. Therefore, in this chapter, we are interested in designing better models to leverage such multi-modal knowledge from the Web.

To address such a multimodal question answering problem, various datasets have been proposed including VQA [70], VQA2.0 [71], CLEVR [72], etc. These different datasets leverage the web images in MSCOCO [73] and Flickr [74], etc. These problems are summarized as visual question answering depicted in Figure 4.1, where the given data is

Figure 4.1: Examples of Visual Question Answering.

an image I and a question Q. The model is tasked with predicting the answer a for the given question Q conditioning on the given image I.

The key challenge of visual question answering, especially in the multi-hop case is how to reason over the visual scenes in the image. Such compositional visual reasoning is a hallmark for human intelligence that endows people with strong problem-solving skills given limited prior knowledge. Neural module networks (NMNs) [75, 76, 77, 78, 79, 80] have been proposed to perform such complex reasoning tasks. NMN requires a set of predefined functions and explicitly encodes each function into unique shallow neural networks called modules, which are composed dynamically to build an instance-specific network for each input question. This approach has high compositionality and interpretability, as each module is designed to accomplish a specific sub-task, and multiple modules can be combined to perform an unseen combination of functions during inference.

However, NMN suffers from two major limitations. 1) **Scalability**: When the complexity of the task increases, the set of functional semantics scales up, so does the number

of neural modules. For example, in the recent GQA dataset [81], a larger set of functions (48 vs 25, see Appendix for details) with varied arity is involved, compared to previous CLEVR dataset [72]. To solve this task with standard NMN framework [75, 77, 80], an increased amount of modules are required to implement for these functions, leading to higher model complexity. 2) **Generalizability**: Since the model is tied to a pre-defined set of functionalities when a new question with unseen functional semantics appears, no existing module can be readily applied to the new semantics, limiting the model's ability to generalize.

In order to enhance NMN for more practical use, we propose Meta Module Network (MMN). As depicted in Figure 4.2, MMN is based on a meta module (a general-purpose neural network), which can take a function recipe (key-value pairs) as input to embed it into continuous vector space and feed it as a side input to instantiate different instance modules. Depending on the specification provided in function recipe, different instance modules are created to accomplish different sub-tasks. As different instance modules inherit the same parameters from the meta module, model complexity remains the same as the function set enlarges. For example, if the recipe has K slots, and each slot takes N values, a compact vector of the recipe can represent up to N^K different functions. This effectively solves the **scalability** issue of NMN. When creating instance modules for specified sub-tasks, the input recipes are encoded into the embedding space for function instantiation. Thus, when an unseen recipe appears, it can be encoded into the embedding space to instantiate a novel instance module based on embedding similarity with previously observed recipes. This metamorphous design effectively overcomes NMN's limitation on **generalizability**.

MMN draws inspiration from Meta Learning [82, 83, 84] as a learning-to-learn approach - instead of learning independent functions to solve different sub-tasks, MMN

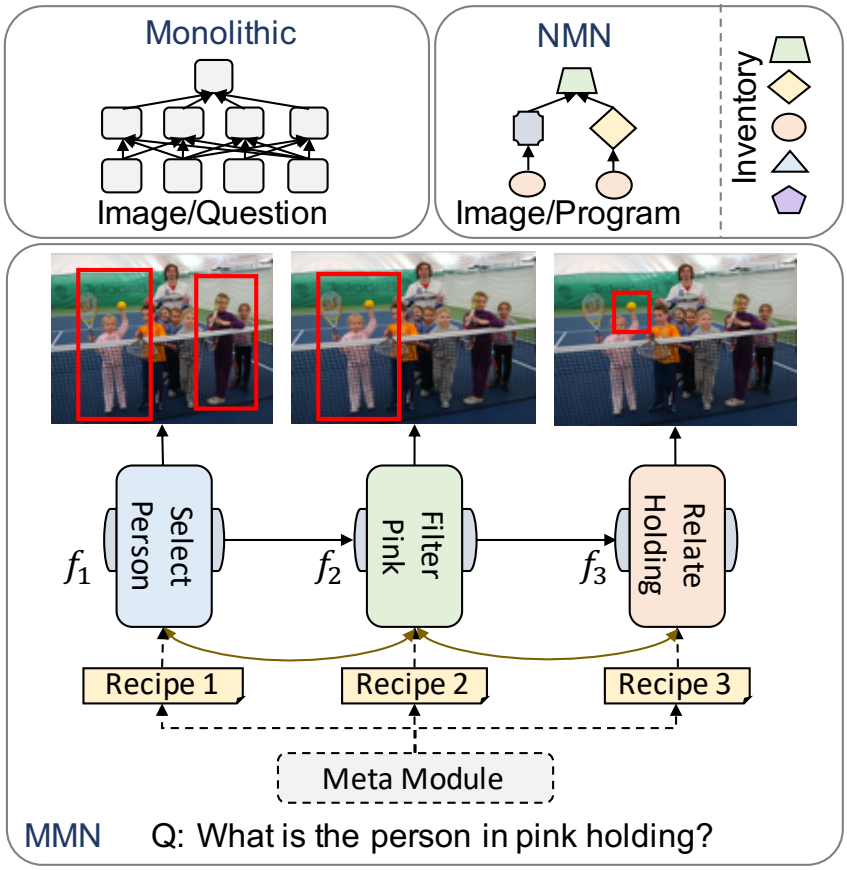

Figure 4.2: Comparison between NMN and MMN for visual reasoning. Neural Module Network (NMN) builds an instance-specific network based on the given question from a pre-defined inventory of neural modules, each module has its independent parameterization. Meta Module Network (MMN) also builds an instance-specific network by instantiating instance modules from the meta module based on the input function recipes (specifications), every instance module has shared parameterization.

learns a meta-function that can generate a function to solve specific sub-task.[1] The learning algorithm of MMN is based on a teacher-student framework to provide module supervision: an accurate "symbolic teacher" first traverses a given scene graph to generate the intermediate outputs for the given functions from specific recipe; the intermediate outputs are then used as guidelines to teach each "student" instance module to accomplish its designated sub-task in the function recipe. The module supervision together

[1]borrowing the concept of *Meta* as in: a book in which a character is writing a book, or a movie in which a character is making a movie, can be described as *Meta*.

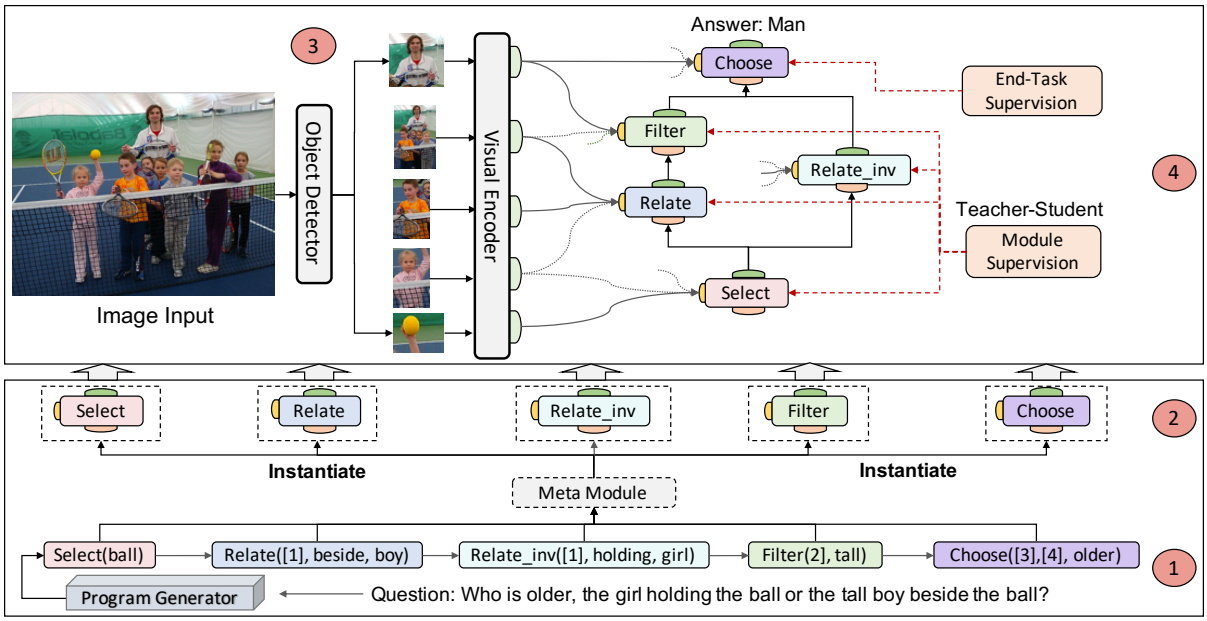

Figure 4.3: The model architecture of MMN: the lower part describes how the question is translated into programs and instantiated into operation-specific modules; the upper part describes the module execution. Circle i denotes the i-th step.

with the original question answering supervision are used jointly to train the model.

The model architecture of MMN is illustrated in Figure 4.3: (1) the coarse-to-fine semantic parser converts an input question into its corresponding program (*i.e.*, a sequence of functions); (2) the meta module is instantiated into different instance modules based on the function recipes of the predicted program, which is composed into an execution graph; (3) the visual encoder encodes the image features that are fed to the instance modules; (4) during training, we provide intermediate module supervision and end-step answer supervision to jointly train all the components.

Our main contributions are summarized as follows. (i) We propose Meta Module Network that effectively extends the scalability and generalizability of NMN for more practical use, allowing it to handle tasks with unseen compositional function from new domain. With a metamorphous meta module learned through teacher-student supervi-

sion, MMN provides great flexibility on model design and model training that cleverly overcomes the rigid hand-crafting of NMN. (*ii*) Experiments conducted on CLEVR and GQA benchmarks demonstrate the scalability of MMN to accommodate larger set of functions. (*iii*) Qualitative visualization on the inferential chain of MMN also demonstrates its superb interpretability and strong transferability.

4.2 Proposed Approach

The visual reasoning task [85] is formulated as follows: given a question Q grounded in an image I, where $Q = \{q_1, \cdots, q_M\}$ with q_i representing the i-th word, the goal is to select an answer $a \in \mathbb{A}$ from a set of possible answers. During training, we are provided with an additional scene graph G for each image I, and a functional program P for each question Q. During inference, scene graphs and programs are not provided.

Figure 4.3 provides an overview of Meta Module Network (MMN), which consists of three components: (*i*) Program Generator (Sec. 4.2.1), which generates a functional program from the input question; (*ii*) Visual Encoder (Sec. 4.2.2), which consists of self-attention and cross-attention layers on top of an object detection model, transforming an input image into object-level feature vectors; (*iii*) Meta Module (Sec. 4.2.3), which can be instantiated to different instance modules to execute the program for answer prediction.

4.2.1 Program Generator

Similar to other programming languages, we define a set of syntax rules for building valid programs and a set of semantics to determine the functionality of each program. Specifically, we define a set of functions $\mathcal{F}$ with their fixed arity $n_f \in \{1, 2, 3, 4\}$ based on the "semantic string" provided in GQA dataset [85]. The definitions for all the functions are provided in the Appendix. The defined functions can be divided into 10 different

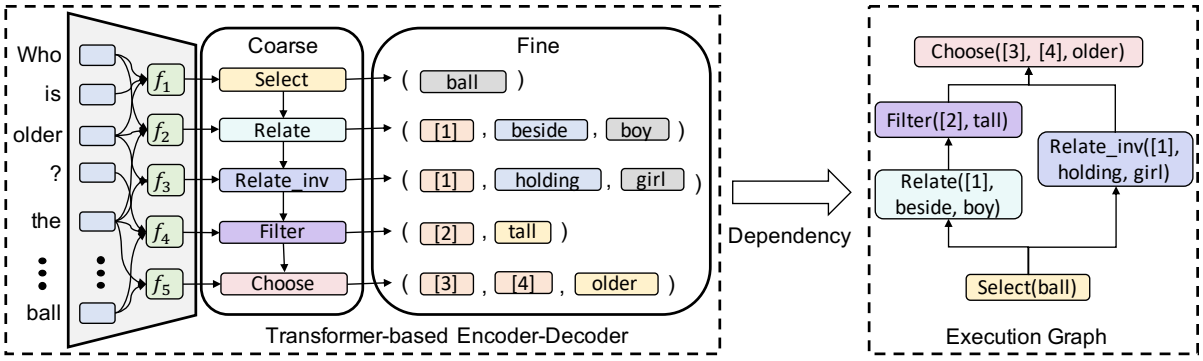

Figure 4.4: Architecture of the coarse-to-fine Program Generator: the left part depicts the coarse-to-fine two-stage generation; the right part depicts the resulting execution graph based on the dependency relationship.

function types (*e.g.*, "*relate*", "*verify*", "*filter*", "*choose*"), and each abstract function type is further implemented with different realizations based on fine-grained functionality (*e.g.*, "*verify*": "*verify_attribute*", "*verify_geometric*", "*verify_relation*", "*verify_color*"), which take different arguments as inputs.

In total, there are 48 different functions defined in GQA environment, which poses great challenges to the scalability in visual reasoning. The returned values of these functions are *List of Objects*, *Boolean*, or *String* (*Object* refers to the detected bounding box, and *String* refers to object name, attributes, relations, etc.) A program P is viewed as a sequence of function calls $f_1, \cdots, f_L$. For example, in Figure 4.4, f_2 is `Relate([1], beside, boy)`, the functionality of which is to find a boy who is beside the objects returned by f_1 : `Select(ball)`. Formally, we call `Relate` the "function name", `[1]` the "dependency" (previous execution results), and `beside, boy` the "arguments". By exploiting the dependency relationship between functions, we build an execution graph, where each node represents a function and each edge denotes an input-output dependency relationship between connected nodes.

In order to generate syntactically plausible programs, we follow [86] and adopt a coarse-to-fine two-stage generation paradigm, as illustrated in Figure 4.4. We first encode

the question as a context vector, and then decode a sketch step by step (the sketch only contains the function name without arguments). Once the sketch is decoded, the arity and types of the decoded functions are determined. For example, after generating "Relate", there are three arguments following this function with the first argument as the dependency. The sketch is thus expanded as "Relate (#1, #2, #3)", where "#i" denotes the i-th unfilled slot. We then apply a fine-grained generator to fill in the slots of dependencies and arguments for the sketch as a concrete program P. During the slot-filling phase, we mask the infeasible tokens at each time step to greatly reduce the search space.

Such a two-stage generation process helps guarantee the plausibility and grammaticality of synthesized programs. For example, if function `Filter` is sketched, we know there are two tokens required to complete the function. The first token should be selected from the dependency set (`[1]`, `[2]`, ...), while the second token should be selected from the attribute set (*e.g.*, `color`, `size`). With these syntactic constraints to shrink the search space, our program synthesizer can achieve a 98.8% execution accuracy (*i.e.*, returning the same result as the ground truth after execution) compared to execution accuracy of 93% of a standard sequence generation model.

4.2.2 Visual Encoder

The visual encoder is based on a pre-trained object detection model [87, 88] that extracts from image I a set of regional features $\mathbf{R} = \{\mathbf{r}_i\}_{i=1}^{N}$, where $\mathbf{r}_i \in \mathbb{R}^{D_v}$, N denotes the number of regions of interest, and D_v denotes the feature dimension. Similar to a Transformer block [89], we first use two self-attention networks, SA_q and SA_r, to encode the question and the visual features as $\hat{\mathbf{Q}} = SA_q(Q, Q; \phi_q)$ and $\hat{\mathbf{R}} = SA_r(\mathbf{R}, \mathbf{R}; \phi_r)$, respectively. $\hat{\mathbf{Q}} \in \mathbb{R}^{M \times D}$, $\hat{\mathbf{R}} \in \mathbb{R}^{N \times D}$, and D is the network's hidden dimension. Based

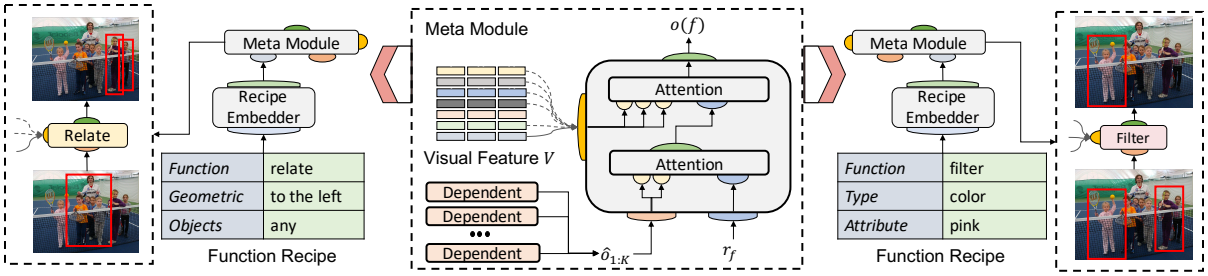

Figure 4.5: Illustration of the instantiation process for `Relate` and `Filter` functions.

on this, a cross-attention network CA is applied to use the question as guidance to refine the visual features into $\mathbf{V} = CA(\hat{\mathbf{R}}, \hat{\mathbf{Q}}; \phi_c) \in \mathbb{R}^{N \times D}$, where $\hat{\mathbf{Q}}$ is used as the query vector, and $\phi = \{\phi_q, \phi_r, \phi_c\}$ denotes all the parameters in the visual encoder. The attended visual features $\mathbf{V}$ will then be fed as the visual input to the meta module, which is detailed in Sec. 4.2.3.

4.2.3 Meta Module

As opposed to having a full inventory of task-specific parameterized modules for different functions as in NMN [76], we design an abstract meta module that can be instantiated into instance modules based on an input function recipe, which is a set of pre-defined key-value pairs specifying the properties of the function. As exemplified in Figure 4.5, when taking recipe `Function:relate; Geometric:to the left` as the input, the Recipe Embedder produces a recipe vector to instantiate the abstract meta module into a "geometric relation" module, which specifically searches for target objects that the current object is to the left of. When taking recipe `Function:filter; Type:color; Attribute:pink` as input, the Embedder will instantiate the meta module into a "filter pink" module, which specifically looks for the objects with pink color in the input objects.

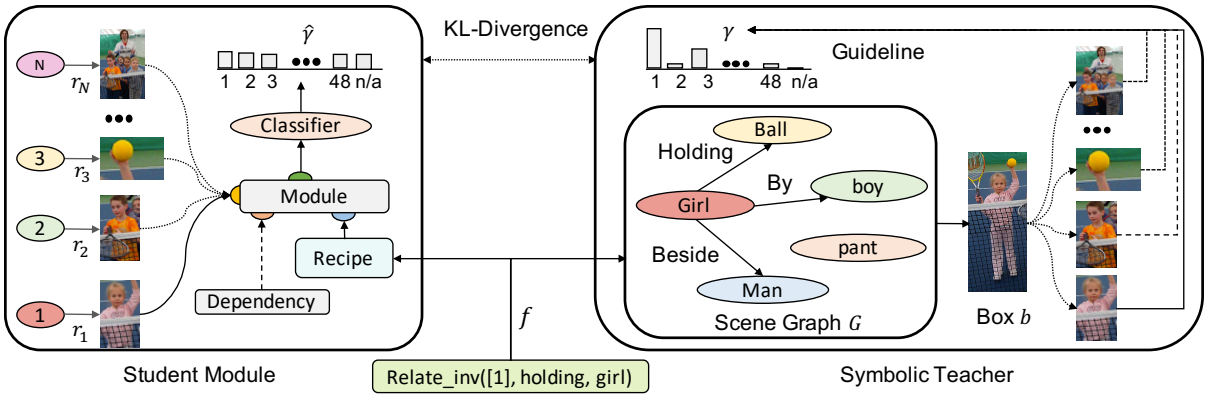

Figure 4.6: Illustration of the Module Supervision process: the symbolic teacher executes the function on the scene graph to obtain the bounding box b, which is then aligned with bounding boxes from the object detection model to compute the supervision guideline.

Two-layered Attention Figure 4.5 demonstrates the computation flow in meta module, which is built upon two-leveled multi-head attention network [89]. A Recipe Embedder encodes a function recipe into a real-valued vector $\mathbf{r}_f \in \mathbb{R}^D$. In the first attention layer, $\mathbf{r}_f$ is fed into an attention network g_d as the query vector to incorporate the output $(\hat{\mathbf{o}}_{1:K})$ of dependent modules. The intermediate output $(\mathbf{o}_d)$ from this attention layer is further fed into a second attention network g_v to incorporate the visual representation $\mathbf{V}$ of the image. The final output is denoted as $g(\mathbf{r}_f, \hat{\mathbf{o}}_{1:K}, \mathbf{V}) = g_v(g_d(\mathbf{r}_f, \hat{\mathbf{o}}_{1:K}), \mathbf{V})$.

Instantiation & Execution The instantiation is accomplished by feeding a function f to the meta module g, which results in a wrapper function $g_f(\hat{\mathbf{o}}_{1:K}, \mathbf{V}; \psi)$ known as instance module (ψ denotes the parameters of the module). Each module g_f outputs $\mathbf{o}(f) \in \mathbb{R}^D$, which acts as the message passed to its neighbor modules. For brevity, we use $\mathbf{o}(f_i)$ to denote the MMN's output at the i-th function f_i. The final output $\mathbf{o}(f_L)$ of function f_L will be fed into a softmax-based classifier for answer prediction. During training, we optimize the parameters ψ (in meta module) and ϕ (in visual encoder) to maximize the likelihood $p_{\phi,\psi}(a|P, Q, \mathbf{R})$ on the training data, where a is the answer, and

$P, Q, \mathbf{R}$ are programs, questions and visual features.

4.2.4 Learning

In order to train the meta module to learn the instantiation process from given function recipes (*i.e.*, how to generate functions), we propose a Teacher-Student framework depicted in Figure 4.6. First, we define a Symbolic Executor as the "Teacher", which can take the input function f and traverse the provided training scene graph to obtain intermediate results (*i.e.*, distribution over the objects on the ground-truth scene graph). The "Teacher" exhibits it as a guideline $\boldsymbol{\gamma}$ for the "Student" instance module g_f to follow.

Symbolic Teacher We first execute the program $P = f_1, \cdots, f_L$ on the ground-truth scene graph G provided in the training data to obtain all the intermediate execution results. According to the function definition (see Appendix for details), the intermediate results are either of type *List of Objects* or *Boolean*. The strategy of representing the results follows: (i) Non-empty *List of Objects*: use the first element's vertexes $[x_1, y_1, x_2, y_2]$; (ii) Empty *List of Objects*: use dummy vertexes $[0, 0, 0, 0]$; (iii) "True" from *Boolean*: use the vertexes from last step; (iv) "False" from *Boolean*: use dummy vertexes as in (ii). Therefore, the intermediate results can be unified in the form of quadruples denoted as b_i. To align these quadruple b_i regions with the regions $\mathbf{R}$ proposed by the object detector from the visual encoder, we compute its overlap against all the regions $r_j \in R$ as $a_{i,j} = \frac{Intersect(b_i, r_j)}{Union(b_i, r_j)}$. Based on whether there exist any overlaps, we handle the following two cases differently:

- If $\sum_j a_{i,j} > 0$, which means that there exists detected bounding boxes overlapping with the b_i, we normalize $a_{i,j}$ over $\mathbf{R}$ to obtain a distribution $\gamma_{i,j} = \frac{a_{i,j}}{\sum_j a_{i,j}}$ and append an extra 0 in the end to obtain $\boldsymbol{\gamma}_i \in \mathbb{R}^{N+1}$.

- If $\sum_j a_{i,j} = 0$, which means no detected bounding box has overlap with b_i (or $b_i = [0, 0, 0, 0]$), we use the one-hot distribution $\boldsymbol{\gamma}_i = [0, \cdots, 0, 1] \in \mathbb{R}^{N+1}$ as the distribution. The last bit represents "No Match".

We call distributions $\gamma_{i,j}$ the guideline from symbolic teacher. Please refer to the right-most part of Figure 4.6 to better understand the computation.

Student Module We propose to demonstrate the guideline distributions $\gamma_{i,j}$ from the symbolic teacher for student instance modules g_f to imitate. Formally, we let each instance module g_f predict its guideline distribution based on its output representation $\mathbf{o}(f_i)$, denoted as $\hat{\boldsymbol{\gamma}}_i = softmax(MLP(\mathbf{o}(f_i)))$. During training, we enforce the instance module's prediction $\hat{\boldsymbol{\gamma}}_i$ to align the guideline distribution $\boldsymbol{\gamma}_i$ by minimizing their KL divergence $KL(\boldsymbol{\gamma}_i || \hat{\boldsymbol{\gamma}}_i)$. This side task is aimed to help the meta module learn the mapping from recipe embedding space $r_f \in \mathbb{R}^D$ to function space $f \in \mathbb{F}$ like a function factory rather than directly learning independent functions f itself. Such a "learning to learn" or meta-learning paradigm gives our model the capability to generalize to unseen sub-tasks encoded in the recipe embedding space.

Joint Optimization Formally, given the quadruple of $(P, Q, \mathbf{R}, a)$ and the pre-computed guideline distribution γ, we propose to add KL divergence to the standard loss function with a balancing factor η:

$$\mathcal{L}(\phi, \psi) = -\log p_{\phi,\psi}(a|P, Q, \mathbf{R}) + \eta \sum_{i=1}^{L-1} KL(\boldsymbol{\gamma}_i || \hat{\boldsymbol{\gamma}}_i)\,. \tag{4.1}$$

The objective jointly provides the module supervision and end-task supervision, the parameters ϕ, ψ of visual encoder and the module network are optimized w.r.t to it.

Model	Cnt	Exist	Cmp Num	Cmp Attr.	Query Attr.	All
NMN [77]	68.5	85.7	84.9	88.7	90.0	83.7
IEP [78]	92.7	97.1	98.7	98.9	98.1	96.9
MAC [81]	97.1	99.5	99.1	99.5	99.5	98.9
NS [90]	99.7	99.9	99.9	99.8	99.8	99.8
NS-CL [80]	98.2	98.8	99.0	99.1	99.3	98.9
MMN	98.2	99.6	99.3	99.5	99.4	99.2

Table 4.1: Comparison of MMN against the state-of-the-art models on CLEVR test set, as reported in their original papers.

4.3 Experiments

In this section, we conduct the following experiments. (*i*) We first evaluate the proposed Meta Module Network on CLEVR datast [72] to preliminarily validate its effectiveness on the synthetic environment. (*ii*) We then evaluate on the GQA v1.1 dataset [85] and compare it with state-of-the-art methods. As GQA is a more realistic testbed to demonstrate the scalability and generalizability of our model, we will focus on it throughout our experiments. (*iii*) We provide visualization of the inferential chains and perform fine-grained error analysis based on that. (*iv*) We design synthesized experiments to quantitatively measure our model's generalization ability towards unseen functional semantics.

To verify the contribution of each component in MMN, we perform several ablation studies. (1) *w/o Module Supervision vs. w/ Module Supervision.* We investigate the influence of module supervision by changing the hyper-parameter η from 0 to 2.0 to see how much influence the module supervision has on the model performance. (2) *w/o Bootstrap vs. w/ Bootstrap.* We investigate the effectiveness of bootstrapping in training to validate whether we could use the large-scale unbalanced split to benefit on the model's performance.

Ablation (1)	Accuracy	Ablation (2)	Accuracy
MCAN	57.4	MMN w/o BS	58.4
MMN ($\eta = 0$)	58.1	MMN w/o FT	56.5
MMN ($\eta = 0.1$)	59.1	MMN + BS (2 ep)	59.2
MMN ($\eta = 0.5$)	**60.4**	MMN + BS (3 ep)	59.9
MMN ($\eta = 1.0$)	60.1	MMN + BS (4 ep)	**60.4**
MMN ($\eta = 2.0$)	59.5	MMN + BS (5 ep)	60.0

Table 4.2: Ablation study on GQA testdev. BS means bootstrapping, FT means fine–tuning; w/o BS: Directly training on the balanced-split; (n ep) means bootstrapped for n epochs.

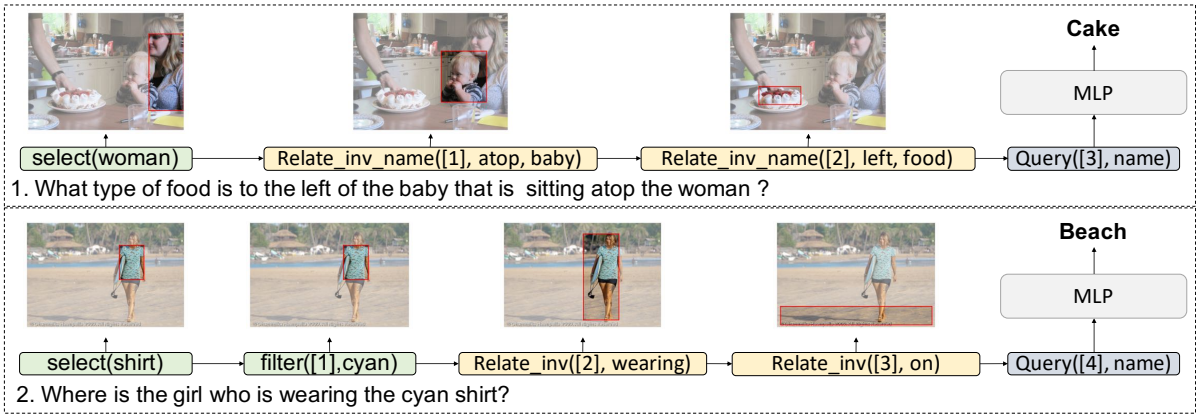

Figure 4.7: Visualization of the inferential chains learned by our model.

We further report the ablation results for the validation split in Table 4.2. From Ablation (1), we observe that without module supervision, our MMN already achieves decent improvement over 6-layered MCAN [91]. Since all the modules have shared parameters, our model has similar parameter size as 1-layered MCAN. The result demonstrates the efficiency of the parameterization in our MMN. By increasing η from 0.1 to 0.5, accuracy steadily improves, which reflects the effectiveness of module supervision. Further increasing the value of η did not improve the performance empirically. From Ablation (2), we observe that bootstrapping is a critical step for MMN, as it explores more data to better regularize functionalities of reasoning modules. Bootstrap for 4 epochs can yield

Function	verify_shape			relate_name		
Methods	NMN	0%(MMN)	100%(MMN)	NMN	0%(MMN)	100%(MMN)
Accuracy	50%	61%	74%	5%	23%	49%
Function	filter_location			choose_name		
Methods	NMN	0%(MMN)	100%(MMN)	NMN	0%(MMN)	100%(MMN)
Accuracy	50%	77%	86%	50%	62%	79%

Table 4.3: Analysis of MMN's generalizability to unseen functions. In NMN, since the unseen function does not have pre-defined module, the performance is close to randomness. In MMN, 0% means without any training instances, 100% means fully–supervision.

better performance in our experiments.

4.3.1 Generalization Experimental Results

Similar to Meta Learning [84], we also evaluate whether our meta module has learned the ability to adapt to unseen sub-tasks. To evaluate such generalization ability, we perform additional experiments, where we held out all the training instances containing `verify_shape, relate_name`, `filter_location, choose_name` to quantitatively measure model's performance on these unseen functions. Standard NMN [76] fails to handle these unseen functions, as it requires training instances for the randomly initialized shallow module network for these unseen functions. In contrast, MMN can generalize the unseen functions from recipe space and exploits the structural similarity with its related functions to infer its semantic functionality. For example, if the training set contains `verify_size` (function: verify, type: size, attr: ?) and `filter_shape` (function: filter, type: shape, attr: ?) functions in the recipes, and instantiated module is capable of inferring the functionality of an unseen but similar function `verify_shape` (function: verify, type:shape, attr: ?) from the recipe embedding space. Table 4.3 shows that the zero-shot accuracy of the proposed meta module is significantly higher than NMN (equivalent to random guess), which demonstrates the generalizability of proposed MMN architec-

ture. Instead of handcrafting a new module every time when new function appears like NMN [76], our MMN is more flexible and extensible for handling growing function sets. Such observation further validates the value of the proposed method to adapt a more challenging environment where we need to handle unknown functions.

4.3.2 Interpretability and Error Analysis

To demonstrate the interpretability of MMN, Figure 4.7 provides some visualization results to show the inferential chain during reasoning. As shown, the model correctly executes the intermediate results and yields the correct final answer. More visualization examples are provided in the Appendix. To better interpret the model's behavior, we also perform quantitative analysis to diagnose the errors in the inferential chain. Here, we held out a small validation set to analyze the execution accuracy of different functions. Our model obtains Recall@1 of 59% and Recall@2 of 73%, which indicates that the object selected by the symbolic teacher has 59% chance of being top-1, and 73% chance as the top-2 by the student model, significantly higher than random-guess Recall@1 of 2%, demonstrating the effectiveness of module supervision.

Furthermore, we conduct a detailed analysis of function-wise execution accuracy to understand the limitation of MMN. We found that most erroneous functions are `relate` and `query`, having 44% and 60% execution accuracy respectively. These errors are mainly related to scene understanding, which suggests that the scene graph model is critical to surpassing NSM [92] on performance. However, this is out of the scope of this chapter and we plan to leave it for future study.

4.4 Related Work

Neural Module Networks By parsing a question into a program and executing the program through dynamically composed neural modules, NMN excels in interpretability and compositionality by design [75, 76, 77, 78, 79, 90, 80, 93]. For example, IEP [77] and N2NMN [78] aims to make the whole model end-to-end trainable via the use of reinforcement learning. Stack-NMN [79] proposes to make soft layout selection so that the whole model is fully differentiable, and Neural-Symbolic VQA [90] proposes to perform completely symbolic reasoning by encoding images into scene graphs. However, its success is mostly restricted to simple datasets with a limited set of functions, whose performance can be surpassed by simpler methods such as relational network [94] and FiLM [95]. Our MMN is a module network in concept, thus possessing high interpretability and compositionality. However, different from traditional NMN, to enhance its scalability and generalizability, MMN uses only a general-purpose meta module for program execution recurrently, which makes MMN inherently a monolithic network, ensuring its strong empirical performance without sacrificing model interpretability.

Monolithic Network Another line of research on visual reasoning is focused on designing monolithic network architecture, such as MFB [96], BAN [97], DCN [98], and MCAN [91]. These black-box models have achieved strong performance on challenging datasets, such as VQA [70, 99] and GQA [85], surpassing the NMN approach. More recently, multimodal pre-training algorithms [100, 101, 102, 103, 104] have been proposed that further lift state of the art on diverse tasks such as VQA [99], NLVR^2 [105], and VCR [106]. They use a unified neural network to learn general-purpose reasoning skills [81], which is more flexible and scalable than NMN. Most monolithic networks for visual reasoning resort to attention mechanism for multimodal fusion [107, 108, 109,

91, 96, 110, 97, 111, 112, 113]. To realize multi-hop reasoning on complex questions, SAN [114], MAC [81] and MuRel [115] models have been proposed. As the monolithic network is not tied to any pre-defined functionality, it has better generalizability to unseen questions. However, since the reasoning procedure is conducted in the feature space, such models usually lack interpretability, or the ability to capture the compositionality in language.

GQA Models GQA was introduced in [85] for real-world visual reasoning. Simple monolithic networks [116], MAC netowrk [81], and language-conditioned graph neural networks [113, 117] have been developed for this task. LXMERT [100], a large-scale pre-trained encoder, has also been tested on this dataset. Recently, Neural State Machine (NSM) [92] proposed to first predict a probabilistic scene graph, then perform multi-hop reasoning over the graph for answer prediction. The scene graph serves as a strong prior to the model. Our model is designed to leverage dense visual features extracted from object detection models, thus orthogonal to NSM and can be enhanced with their scene graph generator once it is publicly available. Different from the aforementioned approaches, MMN also performs explicit multi-hop reasoning based on predicted programs to demonstrate inferred reasoning chain.

Part II

Natural Language Generation over Diverse Web Knowledge

Chapter 5

Generating Logically Faithful Text from Web Table

5.1 Introduction

In the previous chapters, we have talked about natural language understanding, which serves as a cornerstone for building a powerful natural language interface to interact with Web knowledge. Another necessary cornerstone is to enable the model to communicate with human language, which is known as natural language generation problem. The model aims to condition on a specific meaning representation to realize it in the form of natural language. Such generation problem has been extensively studied in lots of previous literature, where the model is tasked with generating the description for the given structured data like tables, knowledge graphs, dialog actions, etc. We summarize the different datasets in Figure 5.1. Here, we mainly focused on table-to-text generation task, where we need to summarize the most salient and interesting claims from the given table. This task has been investigated in previous literature [118, 119, 120, 121], and the existing generation models like GPT-2 [56] can already generate very fluent and coherent

sentences. However, a critical property that is necessary but often overlooked is *fidelity*, i.e., what is generated should be faithful to the underlying data, knowledge, or meaning representation. A line of recent work has started to address the surface-level fidelity issue of natural language generation (NLG) by encouraging the model to learn to reuse the verbatim of certain inputs through copy mechanism [122, 123, 121, 119], structured attention [119], or planning and selection/entity modeling [124, 125]. While shown to be effective, most such methods so far are primarily focused on surface-level realization and simply restate the facts in the underlying data (Figure 5.2).

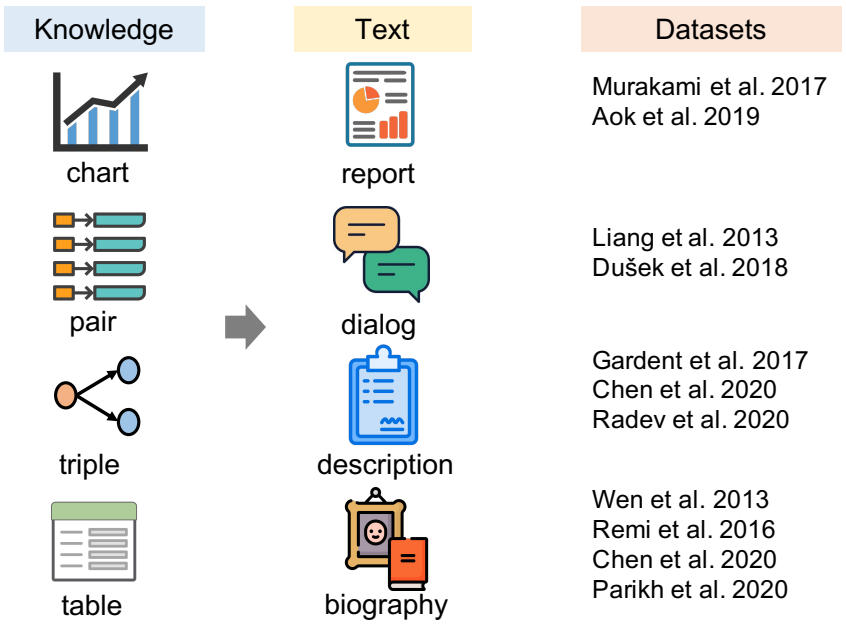

Figure 5.1: Data-to-text Generation from various different structured knowledge.

However, humans have the ability to generalize beyond superficial facts (e.g., "*Canada has got 3 gold medals.*") by inferring and communicating with new statements that can be entailed from these facts (e.g., "*Canada obtained the most gold medals.*"). We believe it is important for NLG models to be able to generalize beyond the superficla facts given to them as well. Therefore, we propose a new task, *logical NLG*, where a model is tasked with generating natural language statements that can be *logically entailed* by

Medal Table from Tournament

Nation	Gold Medal	Silver Medal	Bronze Medal	Sports
Canada	3	1	2	Ice Hockey
Mexico	2	3	1	Baseball
Colombia	1	3	0	Roller Skating

Surface-level Generation
Sentence: Canada has got 3 gold medals in the tournament. **Sentence:** Mexico got 3 silver medals and 1 bronze medal.

Logical Natural Language Generation
Sentence: Canada obtained 1 more gold medal than Mexico. **Sentence**: Canada obtained the most gold medals in the game.

Figure 5.2: Table-to-text generation examples with and without implicit logical inference. Logical NLG requires a generation model to generate natural language statements that can be logically entailed by the facts in the table instead of simply restating certain superficial facts in natural language.

the given data (i.e., the *premises*). The new task requires a model to jointly reason and generate sentences that are consistent both linguistically and logically. Since there are a variety of reasoning/inference tasks such as natural language inference [126] and commonsense reasoning [127], to avoid confusion, this chapter is specifically focused on inferences involving symbolic operations over the given table [24].

To empower research in this direction, we collect a new corpus LogicNLG based on the existing TabFact [63], which brings two major renovations to the existing NLG paradigm: 1) the text involves diversified types of logical inferences including math operations like max/min/sum/add, comparison operations like same/different, and counting operations like total/only. A more detailed description of logical inference is listed in the Appendix. 2) while existing datasets are often restricted to a specific domain such as weather [118], restaurant [120], NBA [121], etc, LogicNLG uses open-domain tables without prior knowledge about their schema. As such, existing methods based on surface-level copying [122, 123, 124] becomes insufficient, so are the existing fidelity evaluation

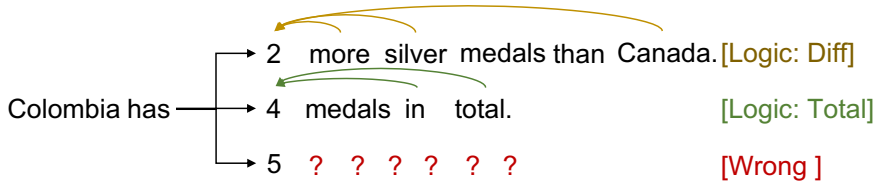

Figure 5.3: When making the decision at the third step, the model needs to foresee the future tokens to ensure logical consistency. There is no back-tracking once the model makes a wrong decision like "5".

based on the surface-level information extraction [121, 128, 129], which extracts surface triples in a certain pre-defined form (i.e. subj-pred-obj, n-gram) and compare them with the surface content given in the knowledge.

Most neural generation models follow a monotonic generation schema from left to right with the current prediction only depending on the preceding words. Logical NLG poses unique challenges to the traditional generation scheme due to the mismatch between *sequence order* and *logical order*. As illustrated in Figure 5.3, the word "2" is derived from the logical inference of 'diff(Silver medal of Colombia, Silver medal of Canada)) $\rightarrow$ 2.' In other words, the logical order of word "2" should be after "more", "silver", and "Canada", while the sequence order of "2" is before those words. Since the monotonic generation scheme is purely based on sequence order while agnostic to logical order, existing NLG models struggle to maintain the fidelity as they cannot model the logical dependency on future tokens. To alleviate such an order mismatch, an NLG model must have the capability to plan ahead for the next few steps before generation. In this context, we believe LogicNLG to be an important testbed to study such a planing/inference ability in generation models [130, 131]. In this chapter, we further propose a non-monotonic coarse-to-fine generation model and show that it is able to alleviate the order mismatch problem and achieve better performance. The contribution of this work is three-fold:

- We propose a new research problem of logical natural language generation, and provide novel metrics to approximately evaluate the logical fidelity of generation

	Vocab	Examples	Tables	Domain	Inference	Schema
WEATHERGOV	394	22.1K	22.1K	Weather	No	Known
WikiBIO	400K	728K	728K	Biography	No	Limited
ROTOWIRE	11.3K	4.9K	4.9K	NBA	Few	Known
LogicNLG	122K	37.0K	7.3K	**Open**	**Rich**	**Unlimited**

Table 5.1: Comparison of LogicNLG against existing NLG datasets in different aspects.

models.

- We justify the mismatch problem between sequence order and logical order of the traditional monotonic generation scheme in logical NLG.
- We conduct comprehensive experiments with state-of-the-art neural generation models under both automatic and human evaluation, which demonstrates the challenges and opportunities for future research on logic NLG.

5.2 Dataset and Problem Definition

Existing NLG datasets [132, 120, 133, 118] are mainly composed of surface-level description over the given records. Though ROTOWIRE [121] involves sporadic inference in the long document, and the inference is restricted to domain-specific knowledge (e.g. double-double, smash, triple-double and other NBA-related terms). Hence, we need a better testbed for studying the proposed problem.

Statistics We construct a dataset based on TabFact [63], which is a table-based fact-checking dataset with rich logical inferences in the annotated statements. Specifically, we took their positive statements (the sentences which are entailed by the knowledge in the table) collected from "complex channel" (required to annotate sentences with logical inference) as our target text. To prevent confusion with the original dataset, we name this

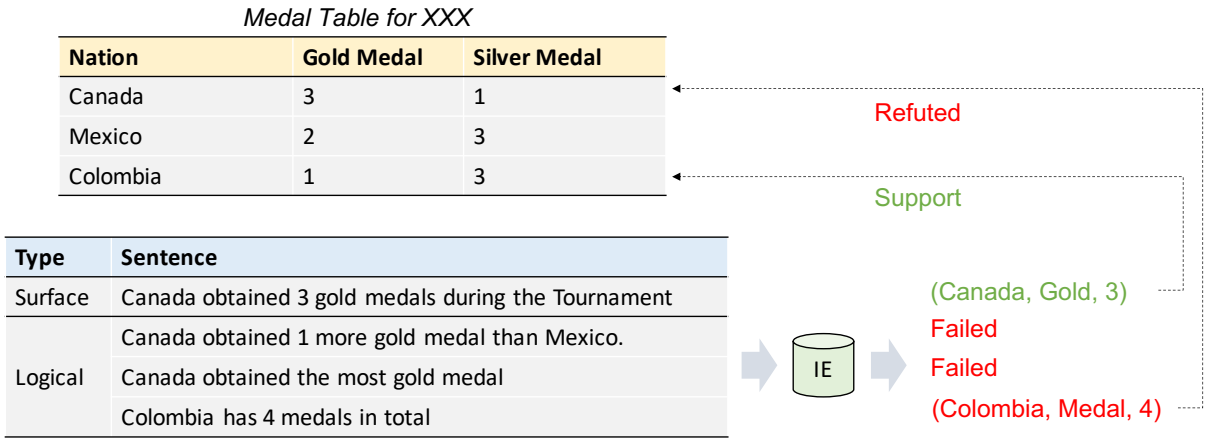

Figure 5.4: Evaluation of surface-level generation vs. logical natural language generation. It suffices to use IE-based evaluation [121, 128] to verify surface-level generation, but it causes either "empty triple" or "false negative" problems to verify logical NLG.

table-to-text dataset as LogicNLG, which contains 28,450 training, 4,260 validation and 4,305 test examples based on 7,392 open-domain tables crawled from Wikipedia. Each table has 5 different examples covering diverse types of logical inference. More detailed statistics and comparisons are listed in Table 5.1. LogicNLG is distinguished from the existing datasets due to:

- It involves very rich logical inference, every annotated sentence involves certain types of inference with minimum domain-specific knowledge. The open-domain characteristic simulates a realistic setting, where we cannot enumerate the possible inference based on the scheme, which poses great challenges to the model's generalization capability.

- It is mainly composed of short sentences with an average length of 11 and a simple syntactic structure, which isolates from other linguistic complexity to focus on the problem of logical inference.

The dataset contains tables with open schema crawled from diversified domains Fig-

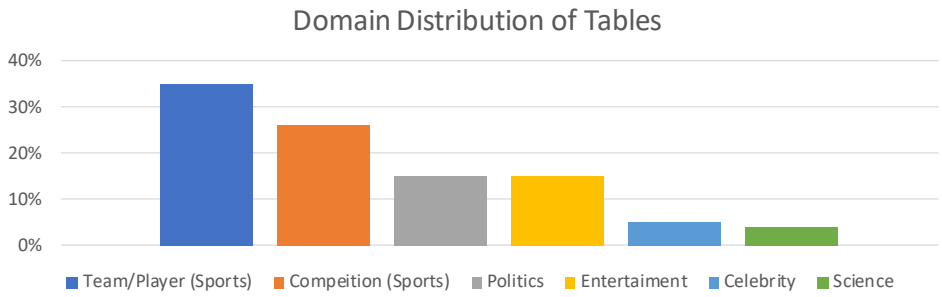

Figure 5.5: The domain distribution of LogicNLG.

ure 5.5. The major categories are sports, politics, and entertainment.

The schema diversity of the tables make the rule-based system infeasible to apply. Besides, most of the tables have very rich numeral records, which provide a great testbed for logical inference.

Problem Definition Here, we formally define our proposed table-to-text generation task. The input is a table $\mathbf{T}$ with its title denoted as a natural language sequence W. The table $\mathbf{T} = \{T_{i,j} | i \leq R_T, j \leq C_T\}$ has R_T rows and C_T columns with the T_{ij} being the content in the (i, j)-th cell. T_{ij} could be a word, a number, a phrase or even a natural language sentence. The annotated statement is a sentence $Y = y_1, y_2, \cdots, y_n$, we aim to train a neural generation model $p(Y|\mathbf{T})$ to generate statement $\hat{Y}$ which are both fluent and logically (numerically) supported by the given table $\mathbf{T}$.

5.3 Automatic Evaluation

In this section, we discuss the evaluation of our proposed NLG task. The fluency evaluation is simply based on the standard metrics like Perplexity [134] and BLEU-1,2,3 [5] based on NLTK [135]. The most challenging problem is to evaluate the *logical fidelity* of the generated sentences, which is also the core problem of this chapter. The

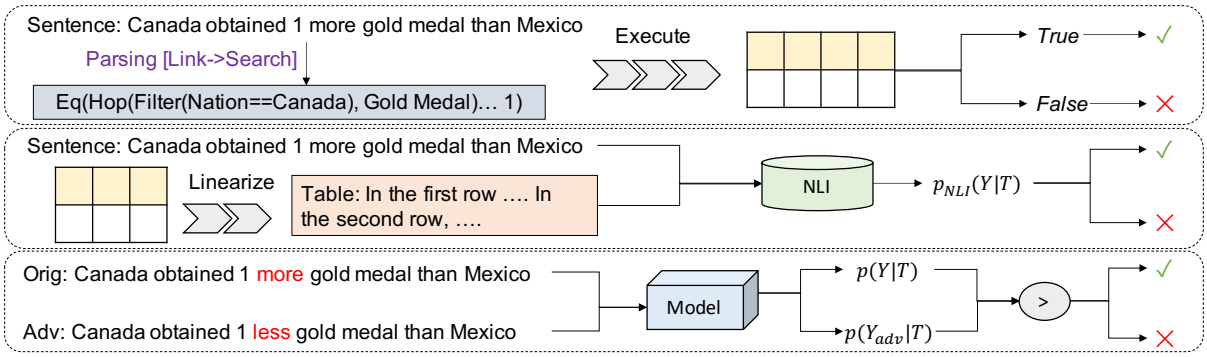

Figure 5.6: The parsing-based and adversarial evaluation to measure model's correctness in logical reasoning.

existing IE-based extractive evaluation [121] leads to two issues as shown in Figure 5.4: 1) Empty Extraction: the sentence can not be formulated as (subject, predicate, object) structure, thus the IE system fail to extract triples for verification. 2) False Negative: the sentence is a logical composition (instead of surface form) of the fact from the table, the IE system cannot match it against the table. For these reasons, we test two approximate automatic metrics:

Parsing-based Evaluation We first propose a model-based evaluation method, which aims to directly extract the meaning representation from the generated sentence and execute it against the table to verify its correctness. Our evaluation is based on weakly-supervised semantic parsing [118, 136], the basic idea is to first link entities and predicates in the sentence, and then use linked entities to perform a breadth-first search to synthesize potential logical forms, finally, a scorer is used to re-rank these logical forms and filter out spurious ones. The logical form returns a binary value of `True` to indicate whether its logic is supported by the knowledge. The basic idea is shown in the upper part of Figure 5.6, the implementation details are in the Appendix. We pre-train the semantic parser f_γ on the training set $(\mathbf{T}, Y) \in D_{train}$ with weakly supervised algorithm, at test time, we

use it to parse a sentence Y into a set of logical forms, which is re-ranked to obtain the highest logical form P_{best}. We compute the ratio of P_{best} returning "true" on D_{test} to approximate model's fidelity.

$$\text{SP-Acc} = \mathop{\mathbb{E}}_{(\mathbf{T},\hat{Y})\in D_{test}} \mathbb{I}(P_{best} \rightarrow True | P_{best} = f_{\gamma}(\hat{Y}))$$

where $\mathbb{I}$ is the indicator function.

NLI-based Evaluation We then propose another model-based evaluation method to complement the parsing-based evaluation (which is sensitive to semantic variation), the basic idea follows [137] to evaluate the entailment score between the table and the generated sentence. The NLI model is based on TableBERT [63], which linearizes the table into textual form and uses it as the evidence for natural language inference. The model is trained with TabFact [63] dataset containing both positive/negative samples. During the evaluation, we use this NLI model to predict the entailment relationship based on the likelihood of $p_{NLI}(Y|T)$. Finally, we compute the ratio of "entailed" to approximate model's fidelity:

$$\text{NLI-Acc} = \mathop{\mathbb{E}}_{(\mathbf{T},\hat{Y})\in D_{test}} \mathbb{I}(p_{NLI}(Y|\mathbf{T}) > 0.5)$$

where $\mathbb{I}$ is the indicator function.

Adversarial Evaluation Adversarial evaluation [138, 139] is used to study the generation model's robustness in logical reasoning. Specifically, we hire human workers from Amazon Mechanical Turk[1] to annotate adversarial examples for the test/validation set by simply changing minimum words to revert the logic of the sentence. Such adversarial

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

examples preserve linguistic components like length and style except the logic-related words to specifically disentangle the generation model's reasoning skill. As drawn in the lower part of Figure 5.6, the original sentence modifies its word "more" into "less" as an adversarial example. There are two principles the workers need to follow to make their jobs accepted: 1) the modified words/phrases should be roughly equally frequent to balance the language prior, for example, the number "1" is better swapped with "2,3" rather than "9999" which rarely appears in the corpus. 2) the perturbation should be diverse enough to cover different aspects of logical reasoning skills. We use the generation model $p(Y|\mathbf{T};\beta)$ to score the original sentence Y and the adversarial sentence Y_{adv}. If the confidence of the original example is higher than its adversarial counterpart, we count it as a successful defense, otherwise as a failed defense. We use the success rate to approximate model's logical reasoning capability.

$$\text{Adv-Acc} = \mathop{\mathbb{E}}_{(\mathbf{T},Y,Y_{adv})\in D_{test}} [\mathbb{I}(p(Y|\mathbf{T}) > p(Y_{adv}|\mathbf{T}))]$$

where $\mathbb{I}$ is the indicator function.

Discussion Both types of metrics have pros and cons, the SP-Acc and NLI-Acc are two metrics unbiased as it measures the peak samples in the model's likelihood, however, both metrics are based on imperfect models and thus their evaluation scores are inaccurate. SP-Acc is more sensitive to number/calculation errors, and NLI-Acc is more sensitive to semantic errors, therefore, we report both of them to help increase the metrics' robustness. In contrast, the adversarial evaluation score is accurate in terms of reflecting the model's reasoning capability on the given samples. However, as the provided samples might not lie in the high-confidence area of the model's distribution, it is biased in reflecting the model's general reasoning capability. Though these fidelity metric models are prone to

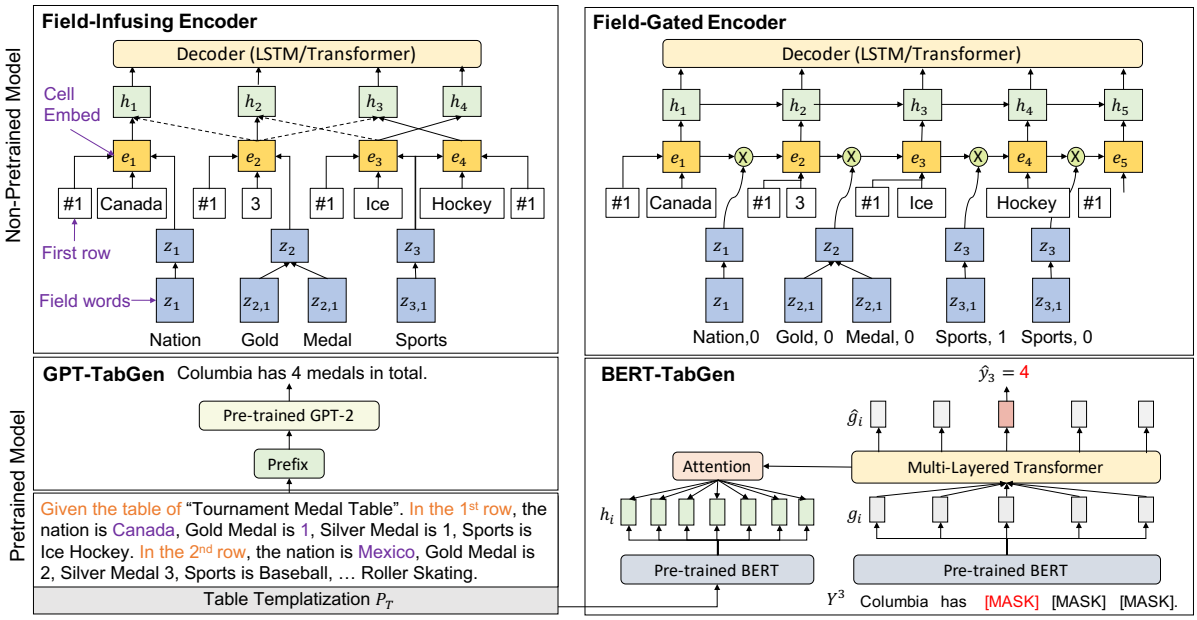

Figure 5.7: The Non-pretrained and Pre-trained generation models, the detailed table is shown in Figure 5.2.

errors, in section 5.6, we show their consistency with human judgment, which reveals their potential to assist human evaluation.

5.4 Baselines

In this section, we design comprehensive baseline models to perform logical NLG. Specifically, we consider the following two cases: non-pretrained models (LSTM/Transformer) with copy mechanism and pre-trained models (GPT-2 and BERT) with sub-word unit. We train these models with three different algorithms: Maximum Likelihood, Adversarial Training, and Reinforcement Learning.

5.4.1 Non-pretrained Models

Here we mainly consider two table encoding methods, namely field-infusing and field-gating. These two methods differ in their strategies to coalesce the field information into cells. After the table is represented as a sequence of vectors, a decoder based on LSTM [140] or Transformer [89] is applied to generate text token by token. The two methods are depicted in the upper part of Figure 5.7:

Field-Infusing This strategy is inspired by [133]. We first use an LSTM [140] to encode the table field text word by word and then use the last output $\mathbf{z_i}$ as field representation. This representation is concatenated with the embedding of row index $\#j$ and word embedding at each cell to obtain a position-aware cell embedding $\mathbf{e_k}$ for each word inside the cell. We stack transformers layers on top of the cell embedding to obtain the table representation as $\mathbf{h_i} \in \mathbb{R}^D$ with D as the dimension.

Field-Gating This strategy is inspired by by [119]. Like the previous strategy, we first use an LSTM [140] to obtain field representation $\mathbf{z_i}$. The field representation is concatenated with ending distance information as the input to an additional field gate built inside the LSTM as suggested in [119], such a field gate is used to control whether the current cell is already encoded. Such a mechanism can help LSTM to identify the boundary between different cells to grasp local information.

5.4.2 Pre-trained Models

To further enhance the fluency and resolve the out-of-vocabulary problem, we use pre-trained language models and finetune them on LogicNLG. Specifically, we consider two models based on GPT-2 [56] and BERT [39], respectively, and name them as GPT-TableGen and BERT-TableGen.

Table Linearization We follow previous work on linearizing knowledge base as natural language [141, 142] to propose "table linearization", which uses template to flatten the table $\mathbf{T}$ as a document $P_T = w_1, \cdots, w_{|T|}$ fed into pre-trained language models to generate statement Y, where we use w_i to denote the i-th word in the generated paragraph P_T and $|T|$ to denote the length of the paragraph (the word w_i is either a table entry or a functional word in the template). As depicted in the left bottom part of Figure 5.7, the original table $\mathbf{T}$ is transformed into a paragraph by horizontally scanning each cell $T_{11} \rightarrow T_{1,C_T} \rightarrow T_{R_T,C_T}$ in the table.

GPT-TabGen we directly feed the paragraph P_T as the input to the pre-trained GPT-2 model and generate the output sentence Y. We finetune the model on LogicNLG by maximizing the likelihood of $p(Y|P_T;\beta)$, with β denoting the parameters of GPT-2 model [56].

BERT-TabGen 1) we encode the linearized paragraph P_T using the pre-trained BERT model into the source representation $\mathbf{h_1}, \cdots, \mathbf{h_{|T|}}$. 2) at the i-th time step, we replace all the words in the groundtruth statement Y after i-th time step by <MASK> token and use BERT to encode the partially masked Y^i as $\mathbf{g_1^i}, \cdots, \mathbf{g_n^i}$. 3) we use an attention layer f_θ to obtain the output hidden states $\hat{\mathbf{g}}_\mathbf{1}^\mathbf{i}, \cdots, \hat{\mathbf{g}}_\mathbf{n}^\mathbf{i}$, where $\hat{\mathbf{g}}_\mathbf{i}^\mathbf{i}$ is used to predict the word $\hat{y}_i$. We jointly optimize β of BERT and θ to maximize the likelihood of generating text Y conditioned on the table and the masked partial sentence. As BERT is a bidirectional model, we need to re-encode the target sentence at each step to get $\mathbf{g_{1:n}^i}$. Therefore, the generation is finished with n passes.

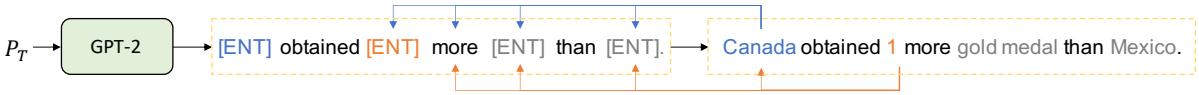

Figure 5.8: Coarse-to-fine generation scheme: first generates a template, and then realize the surface form. It exposes more context to the surface realization model for better capturing logical dependency.

5.4.3 Training

Except for the standard maximum likelihood training, we also use the following training algorithms:

Adversarial Regularization To encourage the model to ground on the table rather than relying on artificial language priors [143], we use an adversarial regularization to enhance the maximum likelihood training. Specifically, we first perform entity resolution to locate all the numbers, count, entities in the sentence and then randomly replace them with entities or numbers appearing in the table $\mathbf{T}$. These perturbed samples Y_{adv} are used as adversarial examples to regularize the model's behavior. Formally, we optimize β to maximize the objective:

$$argmax_{\beta} \log p(Y|\mathbf{T};\beta) - \lambda \log p(Y_{adv}|\mathbf{T};\beta)$$

where λ is the controlling hyper-parameter.

Reinforcement Learning The maximum likelihood training is a fluency-driven objective, which is inconsistent with the goal of logical consistency. To bridge the gap, we view the generation problem from the reinforcement learning perspective to optimize the long-term fidelity. We use the trained semantic parser to assign reward to the policy $p(y_i|y_{1:i-1};\beta)$. At i-th step, the generator will sample different actions y_i and roll-out from $i+1$-th step to produce a full sequence starting from y_i using greedy search. The full

sentence receives a binary score $r(Y, \mathbf{T})$ from the semantic parser as reward. Formally, we optimize the objective:

$$argmax_{\beta} \mathop{\mathbb{E}}_{y_i \sim p(y_i|y_{1:i-1})} [\mathop{\mathbb{E}}_{y_{i+1:n}} [r(y_{1:n}, \mathbf{T})]] \log p(y_i|y_{1:i-1}; \beta)$$

where we only use one trajectory to approximate the inner roll-out expectation for efficiency.

5.5 Coarse-to-Fine Generation

As discussed before, the baseline models follow the monotonic generation scheme and suffer from the mismatch between sequence order and logical order (Figure 5.3). In this section, we propose an imperfect remedy for such a situation based on the coarse-to-fine generation paradigm.

Before plunging into technical details, it is helpful to first realize the resemblance between logical NLG and semantic parsing [86]. Compared to traditional NLG tasks like machine translation and summarization, logical NLG is closer to semantic parsing in the sense that a model may make catastrophic errors that are impossible to be corrected at later steps (Figure 5.3). Therefore, we take inspiration from semantic parsing models [86] that have proven effective in mitigating such errors and propose a coarse-to-fine generation scheme. We break down generation into two phases. In the first phase, the model only generates a template which determines the global logical structure, while in the second phase the model generates the final, grounded sentence conditioned on the template generated in the first phase. As depicted in Figure 5.8, we use the entity linker (Section 5.3) to identify the entities and numbers in the original sentence Y and replace them with placeholder "[ENT]", which we call as the template Y_T. During the genera-

Model	Training	PPL	BLEU-3	SP-Acc	NLI-Acc	Adv-Acc
Field-Gating + LSTM	MLE	27.7	6.9	38.0	56.8	56.2
Field-Gating + Trans	MLE	26.8	8.3	38.5	57.3	58.1
Field-Infusing + LSTM	MLE	27.9	7.1	38.6	57.1	56.9
Field-Infusing + Trans	MLE	26.9	8.4	38.9	57.3	58.2
BERT-TabGen (sm)	MLE	7.5	11.9	42.2	68.1	62.4
GPT-TabGen (sm)	MLE	8.8	12.6	42.1	68.7	62.3
GPT-TabGen (sm)	Adv-Reg	12.1	9.6	40.9	68.5	64.7
GPT-TabGen (sm)	RL	11.3	9.1	**43.1**	67.7	61.9
GPT-C2F (sm)	MLE	-	26.8	42.7	**72.2**	**64.9**
BERT-TabGen (lg)	MLE	6.3	13.5	44.4	73.9	64.0
GPT-TabGen (med)	MLE	6.8	14.2	44.7	74.6	64.3
GPT-TabGen (med)	Adv-Reg	10.1	10.8	44.1	73.0	65.4
GPT-TabGen (med)	RL	10.0	10.0	**45.5**	73.3	63.7
GPT-C2F (med)	MLE	-	**14.6**	45.3	**76.4**	**66.0**

Table 5.2: The experimental results of different models on the test split of LogicNLG, where we split the table into non-pretrained LSTM/Transformer, small pre-trained LM (sm) and medium/large pre-trained LM (med/lg).

tion of GPT-TabGen, instead of directly predicting the final sentence Y, we first predict the template Y_T and then Y. The process is simply realized by maximizing the overall likelihood of $p(\tilde{Y}|\mathbf{T};\beta)$, where $\tilde{Y} = [Y_T; [\text{SEP}]; Y]$.

Unlike template-based or delexicalized generation [4, 144], which uses rigid slot filling prone to grammatic errors, our fine-grained generation has the flexibility to modify the surface form of non-slot words, which alleviates the linguistic coherence problem [145].

By decoupling sentence structure generation and entity grounding, our proposed coarse-to-fine scheme could partially alleviate the mismatch problem. For example, the generation of "Canada" is now aware of "more than" in the latter part of the sentence, which exposes the model to more context than standard monotonic models to help make logically consistent decisions though the dependency on the "1" and "Mexico" is still not captured. The proposed two-step generation could be viewed as the first step towards a fully non-monotonic generation model to solve such mismatch problem.

5.6 Experiments

In this section, we explain the experimental details and then comprehensively report the automatic evaluation of different generation models and training algorithms. Finally, we will conduct detailed human evaluation and error analysis.

5.6.1 Experiment Setup

For the non-pretrained models, we fix the hidden size of both LSTM and transformer to be 256, the transformer is 3-layered with 4 heads, while LSTM is also 3-layered. We use Adam optimizer [146] with a learning rate of 2e-4 to jointly optimize the parameters and keep the model with the best perplexity on the validation set. During test time, we use a greedy search to generate text and calculate the BLEU-1,2,3 scores with the 5 references from the table. For the pre-trained models, we base our implementation on Huggingface's Transformer [147] for both BERT [39] and GPT-2 [56] with subword unit vocabulary of 30K. During linearization, we found that using the whole table compromises the performance greatly, partly due to 1) over-length issue with pre-trained LM, 2) too much irrelevant information input. Therefore, we propose to use partial table as input, specifically, we run entity linking over the sentences to detect the linked columns of the table and only linearize the partial table as input P_T.

Both are finetuned using Adam optimizer [146] with a learning rate of 1e-6. In both adversarial training and reinforcement learning algorithms, we add maximum likelihood objective to stabilize the training, we select the appropriate balancing factor based on the validation Adv-Acc socre. For coarse-to-fine training, we first warm up the model to generate the template sequence and then finetune it on the concatenated full sequence. Model selection is based on the bleu-3 score on validation split.

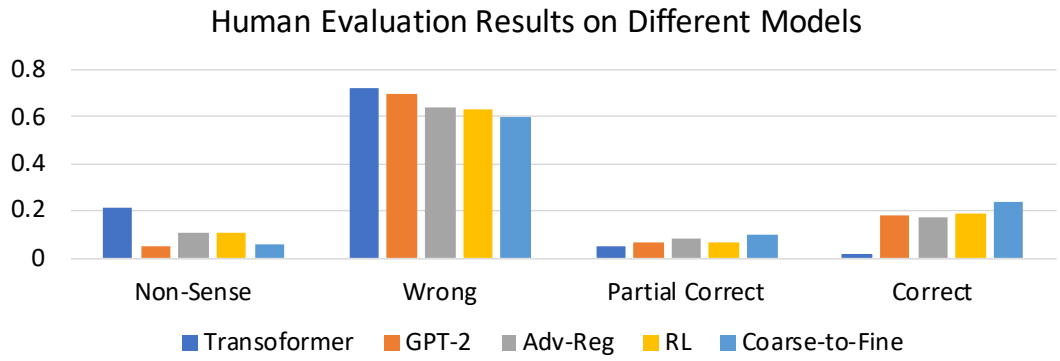

Figure 5.9: The human evaluation results of different models on the sampled sentences.

5.6.2 Experimental Results

We first perform an automatic evaluation to approximately measure the performance of different models and then conduct an in-depth human evaluation to have a better understanding.

Automatic Evaluation: The experimental results are summarized in Table 5.2, where we comprehensively survey different architectures and training algorithms. For the non-pretrained models, we observe that Transformer is slightly better than LSTM and two different table encoding strategies achieve similar results. In contrast, pre-trained models are much better at lowering the perplexity, besides the generated sentences significantly outperform the non-pretrained models in terms of both fluency and fidelity score with GPT-TabGen and BERT-TabGen achieving similar performance. As the BERT-TabGen runs much slower due to multiple passes of decoding, we favor GPT-TabGen in the following experiments. With the adversarial regularization and reinforcement training, the model can only improve the optimized fidelity metric, with the fluency scores dropping significantly. Such phenomena confirm our assumption about the caveats of the monotonic generation paradigm. For the proposed coarse-to-fine generation scheme, as the "[ENT]" tokens are replaced by entity names, which normally contain a phrase like "Feb

2nd". Such n-gram phrase substitution preserves the completeness of entity names and thus leads to higher 2/3/4-gram matches, which translates to higher BLEU-3 and lower BLEU-1 in Table 5.2. The proposed coarse-to-fine generation can yield reasonable improvement over NLI-Acc and Adv-Acc, which demonstrates its advantages of in capturing logical dependency.

Human Evaluation To further investigate the quality of the generated text, we propose to perform human evaluation. Specifically, we sample 200 sentences from different models and distribute them independently to human experts (graduate students from the computer science department) to verify their quality. Specifically, the quality measure is categorized into categories: 1) non-sense: the sentence does not make much sense, which is mainly due to disfluency or repetition problem. 2) wrong: a fluent sentence with wrong logic. 3) partial-correct: the sentence contains more than one fact, at least one of them is correct 4) correct: the high-quality in both fluency and logic correctness. We demonstrate the results in Figure 5.9, from which we observe that pre-training significantly decreases the non-sense proportion. However, the RL and Adv-Reg both harm the fluency and lead to more non-sense sentences. In contrast, the coarse-to-fine model can maintain the non-sense proportion while significantly increasing correct/partial-correct sentences. From human evaluation, even the best performing model can get slightly over 20% of its prediction logically correct, which reflects the challenges of LogicNLG for existing paradigm.

Evaluation Metrics We here analyze the effectiveness of the defined automatic evaluation metrics for fidelity evaluation. For the Parsing-based evaluation and NLI-based evaluation, we use the adversarial set (containing positive/negative sample pairs) to evaluate their consistency with human judges. Parsing-based model only achieves an

accuracy of 60%, while NLI-based model achieves a higher accuracy of 65%. It indicates that the fidelity measurement model is itself a very challenging problem and the existing models are still in a premature stage. Therefore, the exact number of SP-Acc or NLI-Acc cannot reliably reflect the exact proportion of sentences logically entailed by the table. However, we still believe they are informative for model development based on the following reasons: 1) the automatic fidelity scores are quite stable, not sensitive to random initialization or different configurations, 2) when comparing different models (Transformer vs. GPT-2 vs. RL/Adv-Reg vs. Coarse-to-Fine), the trends of different automatic scores are consistent with human evaluation, which indicates its potential in assisting the development of new models.

Fine-grained Analysis To better understand the generation model's reasoning capability in regarding different logical operations, we pick the most frequent 9 operations (definition in the Appendix) and analyze the best model's capability in expressing these different logic. We demonstrate our human evaluation in Figure 5.10 to make the following inspections: 1) the model performs best in justifying the order of different entities (before/after) and relating two entities (both/neither/comparison). 2) the model performs reasonably well at superlative and count operation. 3) the generation model performs much worse in operations like "only, unique". 4) the model is not able to perform mathematical aggregation like average, sum, etc. Overall, the string-based operations are easier than numeric-based operations, how to infuse the numeric knowledge is an open research question to move forward.

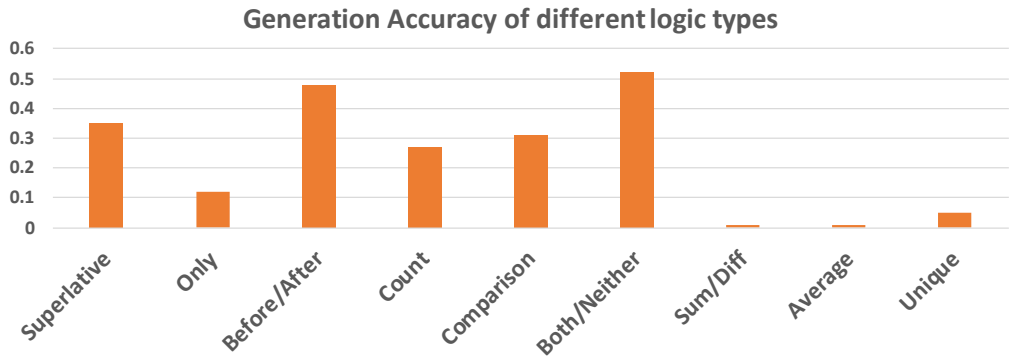

Figure 5.10: The human evaluation results of different models on the sampled sentences.

5.7 Related Work

Natural Language Generation Natural language generation is a long-standing problem [148, 149, 4], which involves generating text from records or data. Recently, many neural-based generation models have been proposed [124, 125, 133, 150] to achieve impressive performance on the existing datasets [132, 118, 133, 120, 121] since the annotated text are mostly surface-level annotation without logical inference. Unlike them, LogicNLG has rich inference, which poses great challenges to existing models and evaluations.

Non-monotonic Generation There have been attempts recently to study the problem of non-monotonic text generation, which aims to teach the generation model to learn the generation order without external supervision [130, 131, 151, 152]. These models have shown to learn rational generation order to approach similar performance as the left-to-right case. These approaches are useful at capturing more sophisticated dependency within the sentence, which provides a plausible direction to pursue in LogicNLG.

Factualness Evaluation Fidelity is an important research topic in generation, In ROTOWIRE [121] and MSCOCO [73], IE-based extractive evaluation [128, 129] are adopted for surface-level matching to replace costly human evaluation. In abstractive

summarization, [153] proposes NER + Relation Classification method to investigate fidelity in generated summarization while [137] proposes to use NLI models to understand the entailment between generated text with the given document. These evaluations are beyond surface-level to study more sophisticated linguistic phenomena like paraphrasing, compression, entailment, inclusion, etc, which are common in summarization tasks.

Chapter 6

Pre-training Data-to-Text Generator with Web Data

6.1 Introduction

The previous chapter has studied the problem of data-to-text generation from Web Tables. The proposed methods are mostly based on pre-trained language models like GPT-2 [56], T5 [154], etc. However, these existing pre-trained language models are pre-trained with pure unstructured textual data while lacking the capability to understand structured data like tables, graphs, charts, etc. Therefore, the structured data needs to be linearized as the input to these language models. With such workaround, their performance of the data-to-text tasks are highly compromised due to the transformation. It's of urgent need to invent a pre-trained generation model to understand and ground on structured data to generate coherent and faith text. In this chapter, we are specifically interested in building such a pre-trained generation model by leveraging distantly supervised (data, text) pairs from the Web without any human annotation.

We hope the pre-trained models could adapt to various of downstream scenarios with

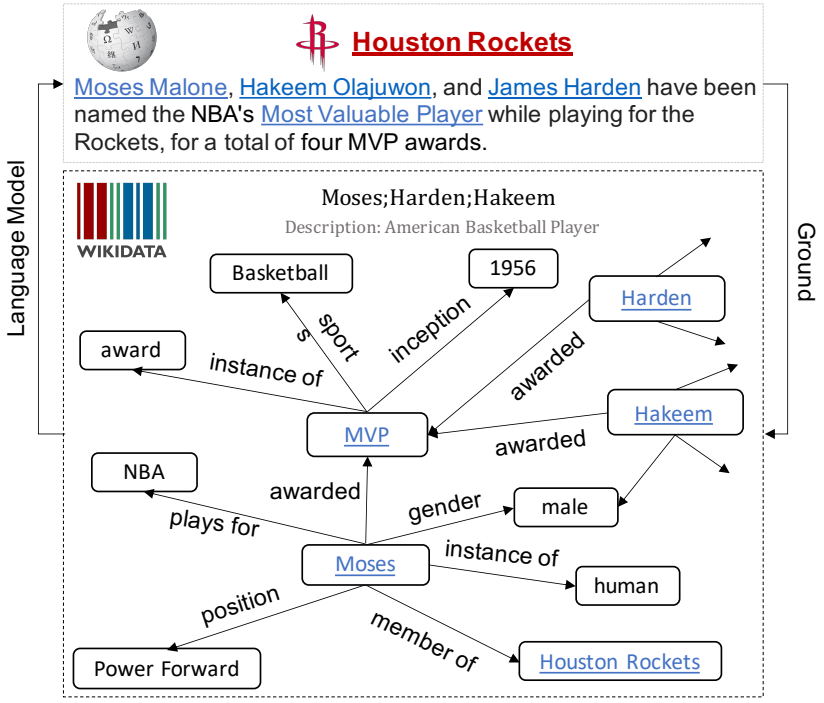

Figure 6.1: An example from the constructed KGTEXT, which pairs a hyperlinked sentence from Wikipedia with a knowledge subgraph from WikiData.

very few or even zero training examples. Our model draws inspiration from the recent wave of pre-trained language model [39, 56, 155] to exploit large-scale unlabeled data from the web for pre-training. The data pairs are constructed through the following procedure. We first crawl sentences with hyperlinks from Wikipedia, and then link the hyperlinked entities to WikiData [156] to find their 1-hop knowledge triples. Finally, we build a subgraph based on the linked triples. Such automatic alignment between knowledge graph and texts provides distant supervision [157] for pre-training but it is bound to be noisy. Therefore, we design a selection strategy and only retain plausible alignments with high semantic overlap. The harvested knowledge-grounded corpus KGTEXT consists of over 1.8M (knowledge subgraph, text) pairs, as depicted in Figure 6.1.

We unify the input of KGTEXT and down-stream data-to-text tasks into a generalized format and design a novel architecture KGPT to encode it. We use KGTEXT

to first pre-train KGPT and then fine-tune it on downstream data-to-text tasks like WebNLG [158], E2ENLG [120] and WikiBio [119]. Experimental results demonstrate KGPT's several advantages: 1) with full down-stream dataset, KGPT can achieve remarkably better performance than known competitive baselines, 2) with zero training, KGPT can still achieve a reasonable score on WebNLG. 3) with a few training instances, KGPT can maintain a high BLEU score while the non-pre-trained baselines only generate gibberish text. A quantitative study shows that our pre-training scheme can reduce annotation costs by roughly 15x to achieve a decent BLEU score of 30. Our contribution is summarized as follows:

- We design a distantly supervised learning algorithm to exploit large-scale unlabeled web text to pre-train data-to-text models.
- The proposed pre-training algorithm can bring significant performance under different settings, especially zero-shot and few-shot scenarios.

6.2 Dataset Construction

The construction process has two stages, namely the crawling stage (downloading data) and the selection stage (filtering data):

6.2.1 Hyperlinked Sentence Crawling

We use English Wikidump[1] as our data source. For each Wikipedia page, we split the whole paragraphs into an array of sentences and then tokenize with the nltk toolkit [159]. We loop through each sentence to keep the sentences with more than 2 Wikipedia anchor links and within the length of 10 and 50. For each candidate sentence, we use its

[1] `https://dumps.wikimedia.org/`

Wikipedia hyperlink to query WikiData [156] and obtain its corresponding entity page[2]. We retrieve the neighboring knowledge triples from these entity pages to construct a local 1-hop graph for each entity. The knowledge triples are divided into two types: 1) the object of the triple is also an entity like '(Roma F.C., country, Italy)', 2) the object of the triple is in plain text like '(Roma F.C., inception, 7 June 1927)'. In the first case, if the object entity also appears in the sentence, we use it as the bridge to build a multi-hop graph like Figure 6.2. After this step, we collected roughly 4 million pairs in the form of (subgraph, sentence) as the candidate for the following step.

6.2.2 Data Selection

We observe that the collected pairs are overly noisy with many sentences totally irrelevant to their paired subgraphs. Apparently, these pairs cannot serve our goal to build a knowledge-grounded language model. Therefore, we propose a data selection step to suppress the noise and filter out the data pairs of our interests. An example is depicted in Figure 6.2, the first sentence does not rely on any information provided by the knowledge graph, while the second sentence has a tight connection to the facts presented in the knowledge graph. Ideally, our proposed strategy should favor the second sentence over the first one.

To achieve this, we propose a simple lexical-based selection strategy to perform data selection. For example, the sentence 'He was born ...' in Figure 6.2 has two query words 'Italy' and 'Germany', we will conduct two rounds of lexical matching. In the first round, we use 'Italy' to query its surrounding neighbors in WikiData to the neighboring unigram, i.e. '(Rome, capital, Europe, Continent, Country, Roma F.C)'. We compute the unigram overlap with the original sentence '(He, was, ...)', which is still 0%. In the second round, we use 'Germany' to do the same computation and calculate the lexical

[2] `https://www.wikidata.org`

overlap, which is still 0%. So the final averaged grounding score of two rounds is 0%. We can follow the same procedure to compute the grounding score for the second sentence in Figure 6.2 with four rounds '(AS Rome, FB, Rome, Italy)'. The grounding score is above 30%, which indicates that the sentence is highly grounded on WikiData subgraph. In this chapter, we use a threshold of 0.13, which selects the top 7M 'good' sentences from the original 12M Wikipedia corpus.

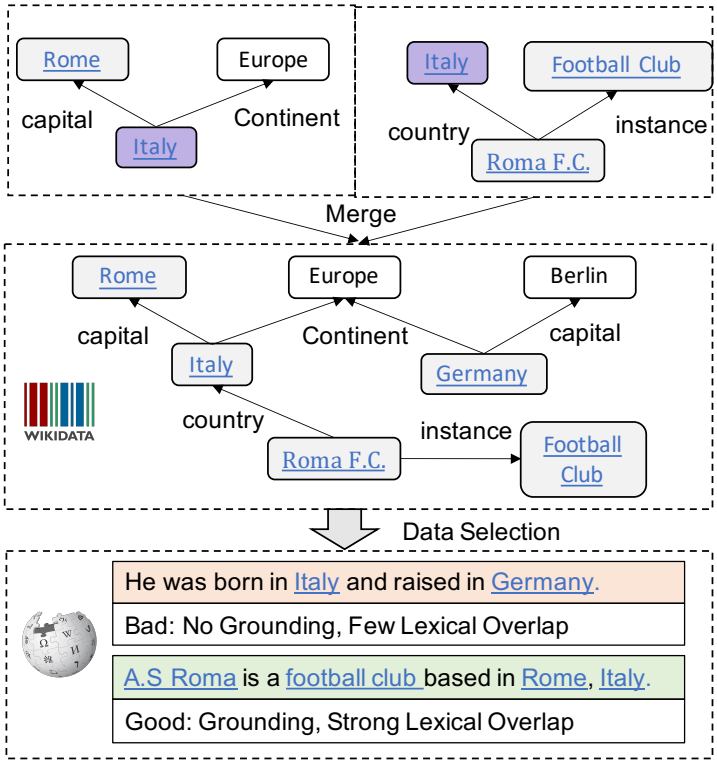

Figure 6.2: Data denoising procedure for the KGTEXT.

After the selection step, we obtain a denoised knowledge-grounded corpus KGTEXT for pre-training. However, there still exist noisy false positives in the corpus, for example, a subgraph contains triple '(Roma F.C., country, Italy)', which is associated with the text 'An Italian player plays for A.S. Roma'. Though the two entities co-occur, they are not meant to describe the fact triple. By applying more strict rules, we can suppress such false positives, but the data capacity could significantly drop consequently. We

experimented with different thresholds to balance noise and data capacity and finally decide on a threshold with an acceptable noise degree. The detailed statistics of the KGTEXT is listed in Table 6.1. We held-out 10,000 sentences for both validation and testing to evaluate the pre-trained model.

#Sent	Length	#Ent	#Pred	#Triple	#Ent/Sent
7M	20.2	1.8M	1210	16M	3.0

Table 6.1: Statistics of collected KGText dataset

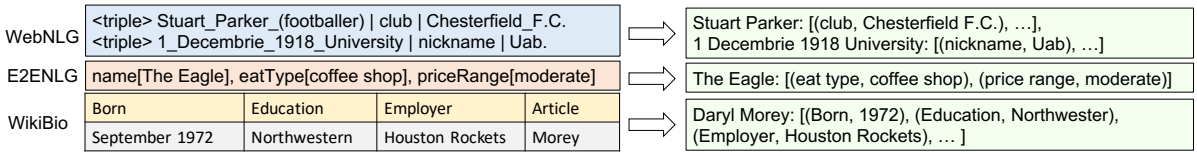

Figure 6.3: The conversion criterion to unify different structured data input into our generalized format.

6.3 Model

We formally define the problem setting and KGPT's architectures in this section.

6.3.1 Problem Setting

In this chapter, we consider inputs from structured data with diverse formats, like knowledge subgraph in KGTEXT, dialog act in E2E [120], RDF triples in WebNLG [158] and tables in WikiBio [133]. Here we unify them into a generalized dictionary format, which uses keys to represent subjects and values to denote the predicate-object pairs following the subject. We showcase the conversion criteria from structured inputs in different data-to-text datasets into our generalized format in Figure 6.3. The generalized

input is denoted as X, and the output is denoted as y. Our model encodes X into a sequence of dense vectors, and then uses the decoder to attend and generate y.

6.3.2 Encoder

The encoder network is crucial to our model to capture the highly structured graph input. We mainly experiment with two types of encoders:

Graph Encoder This encoder is mainly based on graph attention network [160, 161, 162] to explicitly encode the structure information. Specifically, we view each object, predicates, and subjects as the leaf nodes, and add `[ENT]`, `[TRIPLE]` as pseudo nodes for message passing purposes. The built graph is depicted in Figure 6.4.

First of all, we initialize the node representation with the averaged embedding of its subword units. For example, the node 'Moses Malone' has a representation of (E[Mos] + E[es] + E[Ma] + E[lone]) / 4 with E denoting the embedding. After we obtain the initial node representation, we use message propagation to update the node representations based on neighboring information.

In the first layer, we exchange the information between nodes inside a triple, e.g., 'Moses Malone' receives message from siblings 'Gender' and 'Male'. In the second layer, we aggregate information from sub/pred/obj nodes to the `[TRIPLE]` node, e.g., '[TRIPLE1]' receives message from children 'Moses, Gender, Male'. In the third layer, we aggregate the information from different `[TRIPLE]` to the `[ENT]` node. In the fourth layer, we exchange information between different `[ENT]` nodes to enhance cross-entity interactions. Formally, we propose to update the representation of the i-th node $g_i \in \mathbb{R}^D$ with the multi-head attention network, which aggregates information from neighboring

nodes $g_j \in \mathcal{N}_i$ as follows:

$$\begin{aligned} \alpha_j^m &= \frac{e^{(W_Q^m g_i)^T (W_K^m g_j)}}{\sum_{j \in \mathcal{N}_i} e^{(W_Q^m g_i)^T (W_K^m g_j)}} \\ v &= concat[\sum_{j \in \mathcal{N}_i} \alpha_j^m W_v^m (g_j)] \\ \hat{g}_i &= LayerNorm(MLP(v + g_i)) \end{aligned} \tag{6.1}$$

where m denotes the m-th head in the attention layer, $W_Q^m, W_K^m, W_V^m \in \mathbb{R}^{D \times D}$ are the matrices to output query, key, value vectors for m-th head. The attention output v and the residue connection from g_i are fed through the final MLP and LayerNorm to update i-th node representation as $\hat{g}_i$.

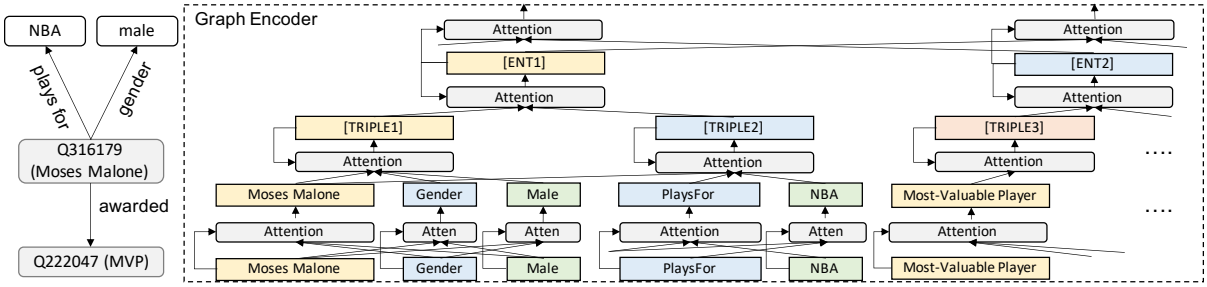

Figure 6.4: Graph Encoder with hierarchical propagation, where we propagate the information from bottom to top.

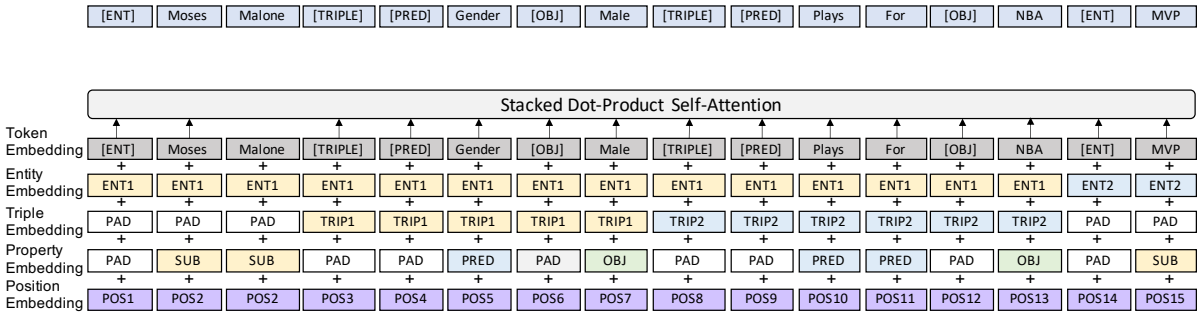

Figure 6.5: Encoding of the knowledge graph as a sequence using special embedding.

The output of graph encoder is denoted as $G \in \mathbb{R}^{n \times D} = \{g_1, \cdots, g_n\}$ with n nodes.

Sequence Encoder This encoder is mainly based on transformer [89] with special embedding as an auxiliary input to infuse the structure information to the sequence model. The concept of special embedding was initially proposed by BERT [39], more recently, it has been adopted by [54] to infuse structural information. We visualize the embedding layer in Figure 6.5, where we leverage additional entity embedding, triple embedding, and property embedding to softly encode the structure of the subgraph as a linearized sequence. For example, the entity embedding can inform the model which entity the current token belongs to, while the triple embedding can indicate which triple the current token belongs to and the property embedding indicates whether the token is a subject, predicate, or a subject. Such an encoding mechanism is designed to softly encode the graph structure into the embedding space for further self-attention. Compared to the graph encoder, the sequence encoder does not enforce the structure as a hard constraint and allows more flexibility for the model to perform cross-triple and cross-entity interactions. Formally, the dot-product self-attention follows the definition of Transformer [89]:

$$
\begin{aligned}
f_{att}(Q, K, V) &= softmax(\frac{QK^T}{\sqrt{D}}V) \\
G_m &= f_{att}(QW_Q^m, KW_K^m, VW_V^m) \\
G &= MLP(Concat(G_1, \cdots, Gm))
\end{aligned}
\tag{6.2}
$$

where Q, K, V are the computed from the input embedding, m represents m-th head and f_{att} is the core attention function, the final output is denoted as $G \in \mathbb{R}^{n \times D}$ with n denoting the sequence length.

6.3.3 Decoder

Our decoder architecture is mainly based on Transformer [89] and copy mechanism [122]. At each decoding time step, the model has a copy gate p_{gen} to select y_i should be generated from the vocabulary $w \in \mathcal{V}$ or copied from the input tokens x:

$$\begin{aligned} \alpha_j &= \frac{e^{o_i^T G_j}}{\sum_{j'} e^{o_i^T G_{j'}}}, \quad p_{gen} = \sigma(MLP(o_i)) \\ P(y_i = w) &= p_{gen} P_{voc}(w) + (1 - p_{gen}) \sum_{j:x_j=w} \alpha_j \end{aligned} \tag{6.3}$$

where o_i is the last layer hidden state of the decoder at i-th time step, α_j is the copy probability over the whole input token sequences x.

6.3.4 Optimization

As we have defined our encoder-decoder model, we will simply represent it as $p_{encdec}(x)$ to output a distribution over word $y_i \in \mathcal{V}$ at the i-th time step. During pre-training, we optimize the log-likelihood function on D_{KGText}. After pre-training, we convert the downstream task's input into the defined dictionary format and denote the dataset as D_{down}, and then further optimize the log-likelihood objective with θ initialized from the pre-training stage.

The pre-train and fine-tuning procedure is displayed in Figure 6.6, where we first use KGText to pre-train KGPT, and then fine-tune with different types of inputs using the standard auto-regressive log-likelihood objective.

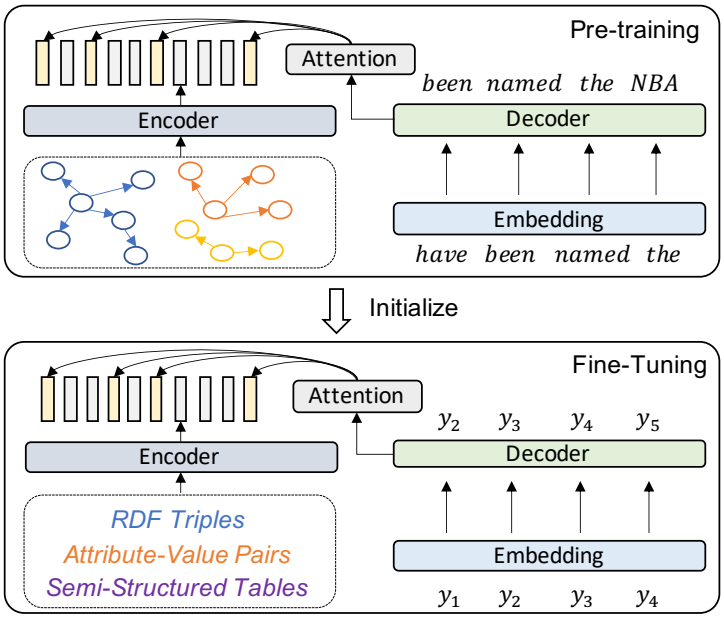

Figure 6.6: Overall pre-training and fine-tuning procedures for KGPT. The down-stream knowledge data formats are converted into the generalized format.

6.4 Experiments

We experiment with three different down-stream tasks, which covers various table-to-text applications to verify the generalization capability of KGPT. Besides the fully supervised learning, we also evaluate zero-shot and few-shot learning.

6.4.1 Datasets

We use WebNLG [158], E2ENLG [120] and WikiBio [133] to evaluate the performance of KGPT. Their basic statistics are listed in Table 6.2. WebNLG and E2ENLG are both crowd-sourced by human annotator while WikiBio is from the Web.

Dataset	Train	Val	Test	Input
WebNLG	34,338	4,313	4,222	RDF Triple
E2ENLG	42,061	4,672	4,693	Dialog Act
WikiBio	582,657	72,831	72,831	Table

Table 6.2: Statistics of different data-to-text datasets

WebNLG This dataset [158] aims to convert RDF triples into a human annotated textual description. We use the recent release 2.0 from GitLab[3]. It contains sets with up to 7 triples each along with one or more references. The number of KB relations modeled in this scenario is potentially large and generation involves solving various subtasks (e.g. lexicalisation and aggregation). As the input RDF triples were modified from the original triples in DBPedia, we first need to check whether there are seen triples in pre-training dataset KGText. We verify that there is zero RDF triple seen during pre-training though 31% entities are seen. Therefore, we can confirm the comparison with other baselines is still fair given no information from test/dev is leaked.

E2ENLG This dataset [120] aims to convert dialog act-based meaning representation into a spoken dialog response. It aims to provide higher-quality training data for end-to-end language generation systems to learn to produce more naturally sounding utterances. In this dataset, each meaning representation is associated with on average with 8.65 different reference utterances.

WikiBio This dataset [133] aims to generate the first sentence of biography description based on a Wikipedia infoboxes table, with each table associated with only one reference. Unlike the previous two human-annotated datasets from different domains, WikiBio is also scraped from Wikipedia. Therefore, we filtered out the instances of KGText from the first paragraph of the biography domain to ensure no overlap or leakage about Wikibio's dev/test set.

[3] `https://gitlab.com/shimorina/webnlg-dataset`

6.4.2 Experimental Setup

We apply the standard GPT-2 [56] tokenizer from Hugginface Github[4] to tokenize the text input, which has a vocabulary of over 50K subword units. We test with both graph encoder and sequence encoder. We set their hidden size to 768 and stack 6 layers for both encoder and decoder with 8 attention heads. During pre-training, we run the model on KGText on 8 Titan RTX GPUs with a batch size of 512 for 15 epochs using Adam [146] optimizer with a learning rate of 1e-4. The pre-training procedure takes roughly 8 days to finish. We use a held-out validation set to select the best checkpoint. During fine-tuning, we use a learning rate of 2e-5.

In our following experiments, we compare with the known best models from different datasets. As none of these models are pre-trained, we also add Template-GPT-2 [13] and Switch-GPT-2 [163] as our pre-trained baselines. Both models apply GPT-2 [56] as the generator to decode description from a table. For the ablation purposes, we list the performance of all non-pre-trained KGPT to see the performance gain brought by pre-training alone. All the best models are selected based on the validation set score, and the numbers are reported in the following tables are for test split. For evaluation, we report the performance with BLEU [5], METEOR [6] and ROUGE-L [7] using e2e-metric[5]. It's worth noting that we perform comprehensive data contamination studies in the following experiments to make sure the pre-training data contains very little overlap with the test split in downstream tasks. We filter out potentially information-leaking pages during the data crawling process.

[4] https://github.com/huggingface/transformers
[5] https://github.com/tuetschek/e2e-metrics

6.4.3 Preliminary Study on KGText

In the preliminary study, we evaluate our pre-trained model's performance on the held-out set of KGTEXT to conduct ablation study over KGPT. Specifically, we investigate 1) which encoding mechanism is better, 2) whether we need copy mechanism or copy supervision. As demonstrated in Table 6.3, we observe that the trivial difference between two encoder designs. With the copy mechanism, KGPT can greatly decrease the perplexity. However, supervising the copy attention does not have much influence on the performance. Therefore, in the following experiments, we will run experiments for both encoding schemes with a copy mechanism without copy loss.

Model	BLEU-4	Perplexity
KGPT-Graph	24.71	4.86
KGPT-Graph + Copy Loss	24.77	4.91
KGPT-Graph w/o Copy	22.69	7.23
KGPT-Seq	24.49	4.95
KGPT-Seq + Copy Loss	24.31	4.93
KGPT-Seq w/o Copy	22.92	7.11

Table 6.3: Ablation Study on held-out set of KGTEXT.

6.4.4 Fully-Supervised Results

We experiment with KGPT under the standard fully-supervised setting to compare its performance with other state-of-the-art algorithms.

WebNLG Challenge We list WebNLG's experimental results in Table 6.4, here we compare with the known models under the unconstrained setting. The baseline models [158] uses sequence-to-sequence attention model [164] as the backbone and propose delexicalization and copy mechanism to enhance model's capability to handle rare items from the input. The GCN model [165] uses graph convolutional neural encoder to en-

code the structured data input. Its implementation is from Github[6]. As can be seen, KGPT without pre-training already achieves better performance than the GCN baseline. With pre-training, the performance is further boosted by 1-2 BLEU-4, which reflects the effectiveness of our method.

Model	BLEU	METEOR	ROUGE
Seq2Seq†	54.0	37.0	64.0
Seq2Seq+Delex†	56.0	39.0	67.0
Seq2Seq+Copy†	61.0	42.0	71.0
GCN	60.80	42.76	71.13
KGPT-Graph w/o Pre-Training	62.30	44.33	73.00
KGPT-Seq w/o Pre-Training	61.79	44.39	72.97
KGPT-Graph w/ Pre-Training	63.84	46.10	74.04
KGPT-Seq w/ Pre-Training	**64.11**	**46.30**	**74.57**

Table 6.4: Experimental results on WebNLG's test set, w/ Training refers to the model with pre-training, otherwise it refers to the model training from scratch. † results are copied from [158].

E2E Challenge We list E2ENLG's experimental results in Table 6.5, here we compare with the state-of-the-art systems on the leaderboard of E2E challenge[7]. These baselines methods are based on neural template model [150], syntax-enhanced algorithms [166], slot alignment [167] and controlling mechanism [168]. As is seen from the table, KGPT can beat the SOTA systems by a remarkable margin. Overall, the improvement brought by pre-training is roughly 0.5-1.0 in terms of BLEU-4, which is less significant than WebNLG. Such a phenomena is understandable given that this dataset contains limited patterns and vocabulary in the input meaning representation, a full training set over 40K instances is more than enough for the generation model to memorize. In the following few-shot experiments, we will show the strength of KGPT to generate high-quality faithful descriptions with only 0.1% of training data.

[6] `https://github.com/diegma/graph-2-text`

[7] `http://www.macs.hw.ac.uk/InteractionLab/E2E/`

Model	BLEU	METEOR	ROUGE
NTemp	55.17	38.75	65.01
TGen	65.93	44.83	68.50
SLUG2SLUG	66.19	44.54	67.72
Adapt	67.37	45.23	70.89
KGPT-Graph w/o Pre-Training	66.47	44.20	67.78
KGPT-Seq w/o Pre-Training	67.67	45.33	70.39
KGPT-Graph w/ Pre-Training	67.87	44.50	70.00
KGPT-Seq w/ Pre-Training	**68.05**	**45.80**	**70.92**

Table 6.5: Experimental results on E2E's test set. NTemp is from [150], TGen is from [166], SLUG2SLUG is from [167] and Adapt is from [168].

WikiBio Dataset We list WikiBio's experimental results in Table 6.6 and compare with models like Table2Seq[169], Order Planning [170], Field Gating [119], Background-KB Attention [171], Hybrid Hierarchical Model [172] trained with multiple auxiliary loss functions. We also train Template-GPT-2 on this dataset to observe pre-trained model's performance. As can be seen from the table, KGPT can achieve better results than the mentioned baseline models. Pre-training can yield an improvement of roughly 0.5 BLEU-4. As this dataset trainin/testing have similar table schema and the large number of training instances already teach the model to memorize the generation patterns, exploiting an external corpus of on par size (1.8M) does not bring a significant boost. So is the template-GPT-2 [13], which performs on par with Field Gating [119]. However, in the few-shot setting, we will show the 25+ BLEU gain brought by pre-training.

6.4.5 Few-Shot Results

The few-shot learning setting aims to study the potential of the proposed pre-training to decrease annotation labor in data-to-text generation tasks. Under this setting, we not only compare with non-pre-trained baselines to observe how pre-training can benefit the model's few-shot learning capability but also compare with other pre-trained LM [163, 13] to see the benefit of KGPT over existing pre-trained LM.

Model	BLEU
Table Neural Language Model [133]	34.70
Table2Seq LSTM [169]	40.26
Order Planning [170]	43.91
Field-Gating LSTM [119]	44.71
KBAtt [171]	44.59
Hierarchical+Auxiliary Loss [172]	45.01
Template-GPT-2	44.67
KGPT-Graph w/o Pre-Training	44.64
KGPT-Seq w/o Pre-Training	44.58
KGPT-Graph w/ Pre-Training	**45.10**
KGPT-Seq w/ Pre-Training	45.06

Table 6.6: Experimental results on WikiBio's test set.

Model	0.5%	1%	5%	10%
Seq2Seq	1.0	2.4	5.2	12.8
Seq2Seq+Delex	4.6	7.6	15.8	23.1
KGPT-Graph w/o Pre-Training	0.6	2.1	5.9	14.4
KGPT-Seq w/o Pre-Training	0.2	1.7	5.1	13.7
Template-GPT-2	8.5	12.1	35.3	41.6
KGPT-Graph w/ Pre-Training	**22.3**	**25.6**	**41.2**	**47.9**
KGPT-Seq w/ Pre-Training	21.1	24.7	40.2	46.5

Table 6.7: Few-shot results on WebNLG's test set.

Model	0.1%	0.5%	1%	5%
TGen	3.6	27.9	35.2	57.3
KGPT-Graph w/o Pre-Training	2.5	26.8	34.1	57.8
KGPT-Seq w/o Pre-Training	3.5	27.3	33.3	57.6
Template-GPT-2	22.5	47.8	53.3	59.9
KGPT-Graph w/ Pre-Training	39.8	**53.3**	**55.1**	**61.5**
KGPT-Seq w/ Pre-Training	**40.2**	53.0	54.1	61.1

Table 6.8: Few-shot results on E2ENLG's's test set.

WebNLG & E2ENLG Dataset In these two datasets, we use 0.1%, 0.5%, 1%, 5%, 10% of training instances to train the model and observe its performance curve in terms of BLEU-4.

For WebNLG challenge, the few-shot situation will pose a lot of unseen entities during test time. From Table 6.7, we can observe that the delexicalization mechanism can

remarkably help with the few-shot situation. However, the improvement brought by delexicalization is much weaker than our proposed pre-training. Under the 5% setting, while the non-pre-trained baselines are only able to generate gibberish text, pre-trained KGPT can maintain a high BLEU score over 40.0 due to its strong generalization ability.

For E2E challenge, the task is comparatively simpler with rather limited items. From Table 6.8, we can observe that TGen [166] is achieving similar performance as our non-pre-trained KGPT, they both perform quite well even under 1% training instances. However, after we further reduce the training samples to roughly 0.1%, the baseline models fail while pre-trained KGPT still maintains a decent BLEU over 40.0.

WikiBio Dataset In this dataset, we adopt the same setting as Switch-GPT-2 [163] and Pivot [173] to use 50, 100, 200 and 500 samples from the training set to train the generation model. From the results in Table 6.9, we observe that KGPT can achieve best scores and outperform both Template-GPT-2 and Switch-GPT-2 under most cases. Though Template-GPT-2 is getting slightly better score with 500 training samples, the overall performance on three datasets are remarkably lower than KGPT, especially under more extreme cases. It demonstrates the advantage of our knowledge-grounded pre-training objective over the naive LM pre-training objective.

Model	50	100	200	500
Field-Infusing	1.3	2.6	3.1	8.2
KGPT-Graph w/o Pre-Training	0.2	1.1	3.8	9.7
KGPT-Seq w/o Pre-Training	0.6	1.7	3.0	8.9
Pivot†	7.0	10.2	16.8	20.3
Switch-GPT-2†	17.2	23.8	25.4	28.6
Template-GPT-2	19.6	25.2	28.8	**30.8**
KGPT-Graph w/ Pre-Training	**24.5**	27.5	28.9	30.1
KGPT-Seq w/ Pre-Training	24.2	**27.6**	**29.1**	30.0

Table 6.9: Few-shot results on Wikibio's test set. † results are copied from [163].

Quantitative Study We further investigate how much sample complexity KGPT can reduce. Specifically, we specify a BLEU-4 score and vary the training data size to observe how much training samples are required to attain the performance. We specify BLEU=30 as our standard and display our results in Table 6.10. We compute the ratio

Model	WebNLG	E2ENLG	WikiBio
KGPT w/o Pre-Training	∼10000	∼300	∼8000
KGPT w/ Pre-Training	∼700	∼20	∼500
Ratio	14x	15x	16x

Table 6.10: Required number of training samples to reach designated BLEU on different dataset.

of sample quantity to characterize the benefits from pre-training. Roughly speaking, pre-training can decrease the sample complexity for training by 15x, which suggests the great reduction rate the annotation cost with pre-trained KGPT to achieve the desired 'promising' performance.

6.4.6 Zero-Shot Results

We further evaluate KGPT's generalization capability under the extreme zero-shot setting and display our results for WebNLG in Table 6.11. As can be seen, all the non-pre-trained baselines and Template-GPT-2 fail under this setting, while KGPT can still manage to generate reasonable outputs and achieve a ROUGE-L score over 30. Given that no input knowledge triples in WebNLG were seen during pre-training, these results reflect KGPT's strong generalization ability to cope with out-of-domain unseen knowledge inputs.

6.4.7 Human Evaluation

We conduct human evaluation to assess the factual accuracy of the generated sentences. Specifically, we sample 100 test samples from WebNLG and observe the model's

Model	BLEU	METEOR	ROUGE
All Baselines	0	0	1.2
Template-GPT-2	0.3	0.5	3.4
KGPT-Graph w/ Pre-Training	13.66	19.17	30.22
KGPT-Seq w/ Pre-Training	13.86	20.15	30.23

Table 6.11: Zero-shot results on WebNLG's test set.

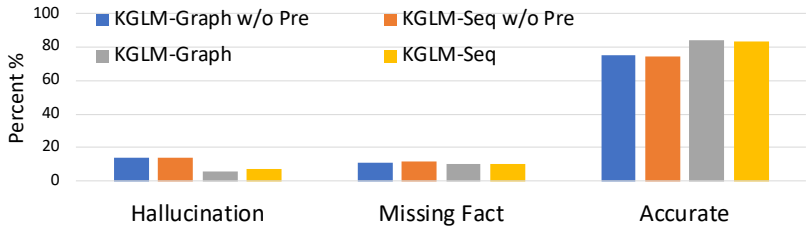

Figure 6.7: Human evaluation of the factual consistency of different models on WebNLG samples.

factual consistency with given fact triples. We use AMT to distribute each generated sentence to four high-quality workers (95% approval rate, 500+ approved jobs) to choose from the three ratings. The majority voted rating is the final rating. We compare four different systems, i.e., non-pre-trained and pre-trained KGPT. Conditioned on the fact triples, we categorize the generated samples into the following categories: 1) hallucinating non-existing facts, 2) missing given facts without hallucination, 3) accurate description of given facts. We visualize the results in Figure 6.7, from which we observe that pre-trained KGPT are less prone to the known hallucination issue and generate more accurate text. The human evaluation suggests that pre-training can enhance the model's understanding over rare entities, thus reducing the over-generation of non-existent facts.

6.5 Related Work

Data-to-Text Generation Data-to-text is a long-standing problem [148, 4], which involves generating natural language surface form from structured data. The traditional

system is primarily built on a template-based algorithm. Recently, with the development of deep learning, attention has been gradually shifted to end-to-end neural generation models, which achieve significant performances on existing large-scale datasets like WebNLG [158], E2ENLG [120], WikiBio [133], ROTOWIRE [174], TOTTO [55], LogicNLG [13], etc. However, these neural generation models are mainly focused on fully supervised learning requiring a huge amount of human annotation for the specific task. This chapter focuses on building a more generalized model architecture, which can adapt to specific tasks well with only a handful of training instances.

Knowledge-Grounded Language Modeling It is of primary importance to ground language models on existing knowledge of various forms. The neural language models [134] have been shown to well capture the co-occurrences of n-grams in the sentences, but falls short to maintain the faithfulness or consistency to world facts. To combat such an issue, different knowledge-grounded language models [175, 176, 177] have been proposed to infuse structured knowledge into the neural language model. These models are mainly focused on enhancing the factualness of unconditional generative models. Inspired by these pioneering studies, we explore the possibility to connect the unconditional generative model with downstream conditional generation tasks. The most straightforward knowledge-intensive conditional generative task is the data-to-text generation, which aims to verbatim given knowledge into lexical format. We demonstrate great potential of the knowledge-grounded pretraining in enhancing the model's factualness on these downstream data-to-text tasks and believe such language models can be applied to broader range of NLP tasks requiring knowledge understanding.

Pre-trained Language Model Recently, the research community has witnessed the remarkable success of pre-training methods in a wide range of NLP tasks [39, 178, 56,

155, 179, 43, 180, 181, 182, 183]. These models trained on millions or billions of data unlabeled data demonstrate unprecedented generalization ability to solve related downstream tasks. However, the existing pre-trained text generation models [56, 180, 183] are initially designed to condition on text input, thus lacking the ability to encode structured inputs. The work closest to our concept is Switch-GPT-2 [163], which fits the pre-trained GPT-2 model as the decoder part to perform table-to-text generation. However, their knowledge encoder is still trained from scratch, which compromises the performance. In this chapter, we follow the existing paradigm to construct an unlabeled web data for LM pre-training.

Chapter 7

Conclusion and Future Work

7.1 Summary

This P.h.D. dissertation has outlined how to build a natural language interface by grounding on the diverse forms of Web knowledge including structured, unstructured, and visual forms. The interface requires two capabilities, 1) grounding on diverse Web knowledge to understand human language input, 2) grounding on diverse Web knowledge to generate natural language response. We use question answering and data-to-text generation as the benchmarks to measure the existing models' performance in these two aspects. We presented different algorithms to enhance models' performance in these two different tasks. However, the problem of building a real-world natural language interface is still far from being done, and this dissertation is simply the beginning of this journey.

Part I is mostly about understanding natural language by grounding on diverse Web knowledge, including structured, unstructured, and multi-modal forms. We resort to various question answering tasks to evaluate models' capabilities to handle different knowledge forms to predict answers. Specifically, we are mostly focused on answering more complex questions, especially multi-hop questions which require multi-step infer-

ence. Our results have shed light on how to utilize different reasoning algorithms with deep neural networks to achieve promising performance. Part II is mostly about natural language generation from diverse Web knowledge, including various types of structured forms. Specifically, we investigate how to build models to generate natural language descriptions conditioning on structured data like graphs and tables, which are pervasive to store world knowledge. Our algorithms have shed light on how to leverage free unlabeled data to bootstrap the existing data-to-text models and enhance its capability to understand structured knowledge inputs. We discovered and proposed several new under-studied research topics, like Hybrid Question Answering [8, 9], Multi-Modal Question Answering [10] and Logical Text Generation [13]. We defined research problems and collected new real-world datasets to enable these research directions. Our datasets can be utilized to properly benchmark the progress of the current models to resolve different use cases.

One core contribution in question answering tasks is our investigation of the knowledge integration problem [8, 9], which is understudied in the recent literature. However, knowledge integration is an essential concept when building a real-world interface like Google Search, etc. We studied different fusion mechanisms to allow mutual interaction and cross attention between different forms, which is able to incorporate knowledge from different forms to perform joint reasoning. Another contribution is the neural symbolic combination in building more powerful reasoning machines [10]. We propose to combine meta-learning into the existing neural module network [75] to greatly improve its generalization skills.

In the text generation tasks, we discovered that the biggest challenge is to maintain the model's fidelity to the given knowledge input. We need to ensure that generated text does not include any hallucination in the generation process. We proposed models to specifically focus on increasing the logical fidelity under table-to-text scenarios [13].

We also propose novel pre-training techniques to handle the zero-shot/few-shot scenarios in data-to-text generation [14]. We expect our approaches to be adopted in real-world applications to have higher impacts on AI research. Therefore, we believe it would be beneficial to summarize the strengths, limitations, and implications of our work as follows.

7.1.1 Reasoning over Hybrid Knowledge

In the proposed datasets, our model is designed to hop between the structured and unstructured forms to aggregate information. However, it is still limited to using the hyperlinks provided in the Wikipedia pages, which is not always realistic in other domains where hyperlink annotations are not provided. For example, in the medical or political domains without annotated links, it's not immediately clear how to reason across different documents. On the other hand, there are many implicit links across different documents, which cannot be plausibly encoded as hyperlinks. Under such more implicit cases, the existing models are incapable to handle such more sophisticated scenarios.

Another disadvantage is the model's weakness to deal with mathematical/numerical reasoning, especially when it comes to different operations like argmax, argmin, sum, diff, etc. As the tabular data normally contains numerical information like numbers, dates, weights, etc, many questions would require performing these operations over the input table. For example, if you ask 'How much older is XX compared to XXX', or 'What is the average score of the team playing in San Antonio in the 2020 NBA Final', etc. Compare or inferring the quantitative relations between different numbers inside the table is necessary to answer these questions. However, the current reader models [49] built with neural networks do not support these mathematical operations. In the future, we might need to consider how to empower the neural models with mathematical reasoning capability.

7.1.2 Reasoning over Multi-Modal Knowledge

The Web contains knowledge in very rich forms, images/videos are pervasive forms to store world knowledge. Thus, it's important for the model to reason under multi-modal scenarios as well. For example, we might ask questions like 'Which color is the jersey of Lebron Jame's home team?' or 'Is Santa Barbara to the north of Los Angeles?', etc. To answer these questions, we normally need to resort to pictures on the Web to obtain information. In our thesis, we mainly experiment with GQA [85] dataset to test the model's capability to reason over visual inputs. To enhance the model's explainability, we aim to develop neural module networks to help us perform compositional reasoning. However, the module networks are still lagging behind the monolithic networks by a huge margin. Such degradation in performance indicates that we are trading the accuracy for better explainability. Moving forward, we need to consider how to further improve neural modules' performance to match the state-of-the-art performance of monolithic networks.

On the other hand, our proposed meta-learning algorithm requires a high-quality scene graph as the training data, which could be unrealistic in different applications. In the future, we need to consider how to better utilize the Web images without extensive annotation to train the module networks.

7.1.3 Fidelity in Text Generation

In data-to-text generation, the problem of fidelity to world knowledge is the primary focus. The existing models are prone to hallucination. For example, if a restaurant doesn't have any open tables in the system but the AI agent still replies to the customer with availability, it might hurt the reputation of the restaurant. How to improve the model's fidelity is a challenging topic because the semantic drift can be very brittle and hard to detect. We proposed the knowledge grounded pre-training algorithm to make

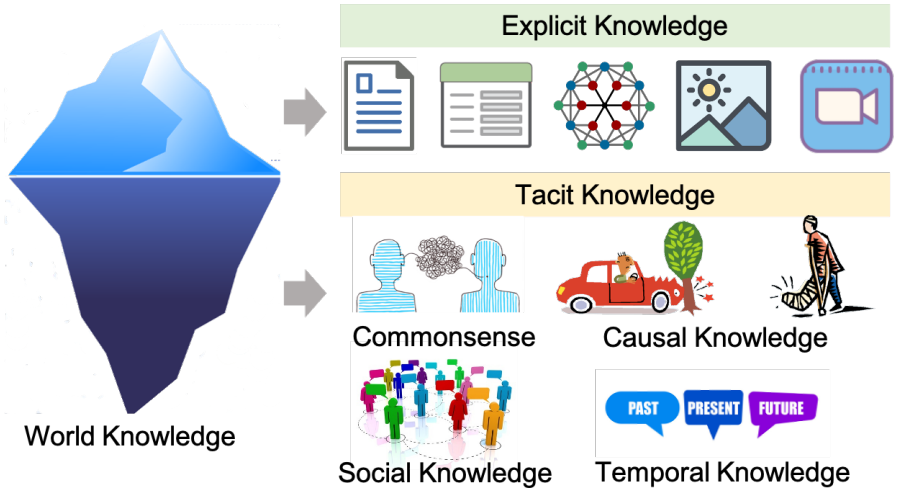

Figure 7.1: The world knowledge iceberg, containing explicit and tacit parts.

the generation model more grounded on the given structured knowledge in KGPT [14]. However, the effectiveness is still limited mainly because the knowledge covered by the WikiData knowledge graph is quite restricted. Moving forward, we need to consider how to increase the coverage of the given structured knowledge and enlarge the pre-training corpus significantly. Potentially, we could utilize some knowledge graph construction, relation extraction methodologies to utilize a broader corpus outside Wikipedia.

7.2 Future Directions

My long-term research goal is to build a better natural language interface that can learn to seek and ground on Web information to communicate with humans. To make the model more adaptable to realistic applications, we still have many problems to address at this moment. With this end goal in mind, I identify the following research directions to pursue next.

7.2.1 Reasoning with Tacit Knowledge

In the previous chapters, we have already covered how to reason over more explicit knowledge like structured, unstructured, or visual data. However, as depicted in Figure 7.1, the world knowledge also consists of tacit parts like commonsense, causal knowledge, social knowledge, etc. The previous approaches by searching over the Web data are not able to cover the tacit knowledge. For example, if the question is 'What will likely happen if I drunk drive?'. By searching over the Web, we won't be able to find any relevant pieces of documents, tables or images containing the answer. However, according to our commonsense, 'drunk driving' will potentially lead to 'car accidents'. Such tacit knowledge is also essential for answering many questions on the Web, or act as an important hop in understanding the question. To leverage such tacit knowledge, retrieval is impossible and we need to find better algorithms to inject such tacit knowledge into the model or the reasoning procedure. In the future, I'm interested in diving deeper to deal with such tacit knowledge and utilize them to help the question-answering tasks.

7.2.2 Reasoning with Adversarial Knowledge

In the previous chapters, we normally assume the given knowledge or data is totally clean and trustworthy. However, there are lot of misinformation on the Web like fake documents, tables, or synthesized images. Such information is either spread out by hackers on purpose or obsolete. The current models don't have the capability to deal with such adversarial knowledge during retrieval or reading to distinguish their falsity. In order to build trustworthy interface models to communicate with humans, we need to better deal with such adversarial knowledge on the Web. In order to resolve such issues, we can use fake detection models to have helped us filter such adversarial knowledge. In the future, I'm interested in investigating deeper to develop models to handle such

adversarial scenarios.

Bibliography

[1] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, *Squad: 100,000+ questions for machine comprehension of text*, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

[2] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, *Glue: A multi-task benchmark and analysis platform for natural language understanding*, *EMNLP 2018* (2018) 353.

[3] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, *Superglue: A stickier benchmark for general-purpose language understanding systems*, *arXiv preprint arXiv:1905.00537* (2019).

[4] E. Reiter and R. Dale, *Building applied natural language generation systems*, *Natural Language Engineering* **3** (1997), no. 1 57–87.

[5] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, *Bleu: a method for automatic evaluation of machine translation*, in *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318, Association for Computational Linguistics, 2002.

[6] S. Banerjee and A. Lavie, *Meteor: An automatic metric for mt evaluation with improved correlation with human judgments*, in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72, 2005.

[7] C.-Y. Lin, *ROUGE: A package for automatic evaluation of summaries*, in *Text Summarization Branches Out*, (Barcelona, Spain), pp. 74–81, Association for Computational Linguistics, July, 2004.

[8] W. Chen, H. Zha, Z. Chen, W. Xiong, H. Wang, and W. Wang, *Hybridqa: A dataset of multi-hop question answering over tabular and textual data*, *Proceedings of Findings of EMNLP 2020* (2020).

[9] W. Chen, M.-W. Chang, E. Schlinger, W. Wang, and W. W. Cohen, *Open question answering over tables and text*, *International Conference on Learning Representations* (2021).

[10] W. Chen, Z. Gan, L. Li, Y. Cheng, W. Wang, and J. Liu, *Meta module network for compositional visual reasoning*, in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 655–664, 2021.

[11] W. Chen, H. Wang, J. Chen, Y. Zhang, H. Wang, S. Li, X. Zhou, and W. Y. Wang, *Tabfact: A large-scale dataset for table-based fact verification*, *International Conference on Learning Representations (ICLR)* (2020).

[12] L. Pan, W. Chen, W. Xiong, M.-Y. Kan, and W. Wang, *Zero-shot fact verification by claim generation*, in *Proceedings of the ACL 2021*, 2021.

[13] W. Chen, J. Chen, Y. Su, Z. Chen, and W. Y. Wang, *Logical natural language generation from open-domain tables*, *ACL 2020* (2020).

[14] W. Chen, Y. Su, X. Yan, and W. Y. Wang, *KGPT: Knowledge-grounded pre-training for data-to-text generation*, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Online), pp. 8635–8648, Association for Computational Linguistics, Nov., 2020.

[15] W. Chen, J. Chen, P. Qin, X. Yan, and W. Y. Wang, *Semantically conditioned dialog response generation via hierarchical disentangled self-attention*, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3696–3709, 2019.

[16] X. Wang, W. Chen, J. Wu, Y.-F. Wang, and W. Y. Wang, *Video captioning via hierarchical reinforcement learning*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4213–4222, 2018.

[17] X. Wang, W. Chen, Y.-F. Wang, and W. Y. Wang, *No metrics are perfect: Adversarial reward learning for visual storytelling*, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 899–909, 2018.

[18] P. Rajpurkar, R. Jia, and P. Liang, *Know what you don't know: Unanswerable questions for squad*, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789, 2018.

[19] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, *Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension*, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

[20] J. Berant, A. Chou, R. Frostig, and P. Liang, *Semantic parsing on freebase from question-answer pairs*, in *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1533–1544, 2013.

[21] W.-t. Yih, M.-W. Chang, X. He, and J. 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)*, pp. 1321–1331, 2015.

[22] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh, *The value of semantic parse labeling for knowledge base question answering*, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 201–206, 2016.

[23] A. Talmor and J. Berant, *The web as a knowledge-base for answering complex questions*, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 641–651, 2018.

[24] P. Pasupat and P. Liang, *Compositional semantic parsing on semi-structured tables*, 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)*, pp. 1470–1480, 2015.

[25] V. Zhong, C. Xiong, and R. Socher, *Seq2sql: Generating structured queries from natural language using reinforcement learning*, *arXiv preprint arXiv:1709.00103* (2017).

[26] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, *et. al.*, *Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task*, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3911–3921, 2018.

[27] H. Sun, T. Bedrax-Weiss, and W. Cohen, *Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text*, 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)*, pp. 2380–2390, 2019.

[28] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. 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*, pp. 4231–4242, 2018.

[29] W. Xiong, M. Yu, S. Chang, X. Guo, and W. Y. Wang, *Improving question answering over incomplete kbs with knowledge-aware reader*, in *Proceedings of the*

57th Annual Meeting of the Association for Computational Linguistics, pp. 4258–4264, 2019.

[30] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song, *Variational reasoning for question answering with knowledge graph*, in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[31] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner, *Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs*, 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)*, pp. 2368–2378, 2019.

[32] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. Cohen, R. Salakhutdinov, and C. D. Manning, *Hotpotqa: A dataset for diverse, explainable multi-hop question answering*, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018.

[33] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, *et. al.*, *Natural questions: a benchmark for question answering research*, *Transactions of the Association for Computational Linguistics* **7** (2019) 453–466.

[34] C. S. Bhagavatula, T. Noraset, and D. Downey, *Methods for exploring and mining tables on wikipedia*, in *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, pp. 18–26, ACM, 2013.

[35] D. Kaushik and Z. C. Lipton, *How much reading does reading comprehension require? a critical investigation of popular benchmarks*, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 5010–5015, 2018.

[36] J. Chen and G. Durrett, *Understanding dataset design choices for multi-hop reasoning*, 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)*, pp. 4026–4032, 2019.

[37] C. Clark, M. Yatskar, and L. Zettlemoyer, *Don't take the easy way out: Ensemble based methods for avoiding known dataset biases*, *arXiv preprint arXiv:1909.03683* (2019).

[38] X. Xu, C. Liu, and D. Song, *Sqlnet: Generating structured queries from natural language without reinforcement learning*, *arXiv preprint arXiv:1711.04436* (2017).

[39] J. Devlin, M.-W. Chang, K. Lee, and K. 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)*, pp. 4171–4186, 2019.

[40] D. Chen, A. Fisch, J. Weston, and A. 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)*, pp. 1870–1879, 2017.

[41] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, *ICLR 2019* (2019).

[42] M. Dunn, L. Sagun, M. Higgins, V. U. Guney, V. Cirik, and K. Cho, *Searchqa: A new q&a dataset augmented with context from a search engine*, *arXiv preprint arXiv:1704.05179* (2017).

[43] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, *Roberta: A robustly optimized bert pretraining approach*, *arXiv preprint arXiv:1907.11692* (2019).

[44] H. Sun, T. Bedrax-Weiss, and W. Cohen, *PullNet: Open domain question answering with iterative retrieval on knowledge bases and text*, 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)*, (Hong Kong, China), pp. 2380–2390, Association for Computational Linguistics, Nov., 2019.

[45] P. Qi, X. Lin, L. Mehr, Z. Wang, and C. D. Manning, *Answering complex open-domain questions through iterative query 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)*, pp. 2590–2602, 2019.

[46] S. Min, D. Chen, L. Zettlemoyer, and H. Hajishirzi, *Knowledge guided text retrieval and reading for open domain question answering*, *arXiv preprint arXiv:1911.03868* (2019).

[47] M. Ding, C. Zhou, Q. Chen, H. Yang, and J. Tang, *Cognitive graph for multi-hop reading comprehension at scale*, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2694–2703, 2019.

[48] A. Asai, K. Hashimoto, H. Hajishirzi, R. Socher, and C. Xiong, *Learning to retrieve reasoning paths over wikipedia graph for question answering*, in *International Conference on Learning Representations*, 2019.

[49] J. Ainslie, S. Ontanon, C. Alberti, P. Pham, A. Ravula, and S. Sanghai, *Etc: Encoding long and structured data in transformers*, *Proceedings of EMNLP 2020* (2020).

[50] M. Zaheer, G. Guruganesh, A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, *et. al.*, *Big bird: Transformers for longer sequences*, *arXiv preprint arXiv:2007.14062* (2020).

[51] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, *Signature verification using a" siamese" time delay neural network*, in *Advances in neural information processing systems*, pp. 737–744, 1994.

[52] K. Lee, M.-W. Chang, and K. Toutanova, *Latent retrieval for weakly supervised open domain question answering*, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 6086–6096, 2019.

[53] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, *Realm: Retrieval-augmented language model pre-training*, *Proceedings of ICML 2020* (2020).

[54] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. M. Eisenschlos, *Tapas: Weakly supervised table parsing via pre-training*, *ACL 2020* (2020).

[55] A. P. Parikh, X. Wang, S. Gehrmann, M. Faruqui, B. Dhingra, D. Yang, and D. Das, *Totto: A controlled table-to-text generation dataset*, *arXiv preprint arXiv:2004.14373* (2020).

[56] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, *Language models are unsupervised multitask learners*, *OpenAI Blog* **1** (2019), no. 8 9.

[57] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et. al.*, *Tensorflow: A system for large-scale machine learning*, in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283, 2016.

[58] C. Zhao, C. Xiong, C. Rosset, X. Song, P. Bennett, and S. Tiwary, *Transformer-xh: Multi-evidence reasoning with extra hop attention*, in *International Conference on Learning Representations*, 2019.

[59] Y. Nie, S. Wang, and M. Bansal, *Revealing the importance of semantic retrieval for machine reading at scale*, 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)*, pp. 2553–2566, 2019.

[60] B. Dhingra, M. Zaheer, V. Balachandran, G. Neubig, R. Salakhutdinov, and W. W. Cohen, *Differentiable reasoning over a virtual knowledge base*, in *International Conference on Learning Representations*, 2019.

[61] H. Sun, H. Ma, X. He, W.-t. Yih, Y. Su, and X. Yan, *Table cell search for question answering*, in *Proceedings of the 25th International Conference on World Wide Web*, pp. 771–782, 2016.

[62] K. Chakrabarti, Z. Chen, S. Shakeri, and G. Cao, *Open domain question answering using web tables*, *arXiv preprint arXiv:2001.03272* (2020).

[63] W. Chen, H. Wang, J. Chen, Y. Zhang, H. Wang, S. Li, X. Zhou, and W. Y. Wang, *Tabfact: A large-scale dataset for table-based fact verification*, in *International Conference on Learning Representations*, 2019.

[64] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel, *Tabert: Pretraining for joint understanding of textual and tabular data*, *ACL 2020* (2020).

[65] R. Child, S. Gray, A. Radford, and I. Sutskever, *Generating long sequences with sparse transformers*, *arXiv preprint arXiv:1904.10509* (2019).

[66] N. Kitaev, Ł. Kaiser, and A. Levskaya, *Reformer: The efficient transformer*, *ICLR* (2020).

[67] A. Roy, M. Saffar, A. Vaswani, and D. Grangier, *Efficient content-based sparse attention with routing transformers*, *arXiv preprint arXiv:2003.05997* (2020).

[68] I. Beltagy, M. E. Peters, and A. Cohan, *Longformer: The long-document transformer*, *arXiv preprint arXiv:2004.05150* (2020).

[69] G. Izacard and E. Grave, *Leveraging passage retrieval with generative models for open domain question answering*, *arXiv preprint arXiv:2007.01282* (2020).

[70] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. Lawrence Zitnick, and D. Parikh, *Vqa: Visual question answering*, in *ICCV*, 2015.

[71] P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, and D. Parikh, *Yin and yang: Balancing and answering binary visual questions*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5014–5022, 2016.

[72] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick, *Clevr: A diagnostic dataset for compositional language and elementary visual reasoning*, in *CVPR*, 2017.

[73] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, *Microsoft coco: Common objects in context*, in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[74] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier, *From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions*, *Transactions of the Association for Computational Linguistics* **2** (2014) 67–78.

[75] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, *Learning to compose neural networks for question answering*, in *NAACL*, 2016.

[76] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, *Neural module networks*, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 39–48, 2016.

[77] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko, *Learning to reason: End-to-end module networks for visual question answering*, in *ICCV*, 2017.

[78] J. Johnson, B. Hariharan, L. van der Maaten, J. Hoffman, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick, *Inferring and executing programs for visual reasoning*, in *ICCV*, 2017.

[79] R. Hu, J. Andreas, T. Darrell, and K. Saenko, *Explainable neural computation via stack neural module networks*, in *ECCV*, 2018.

[80] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu, *The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision*, in *ICLR*, 2019.

[81] D. A. Hudson and C. D. Manning, *Compositional attention networks for machine reasoning*, in *ICLR*, 2018.

[82] J. Schmidhuber, *On learning how to learn learning strategies*, *Citeseer* (1995).

[83] S. Ravi and H. Larochelle, *Optimization as a model for few-shot learning*, *ICLR* (2017).

[84] C. Finn, P. Abbeel, and S. Levine, *Model-agnostic meta-learning for fast adaptation of deep networks*, in *ICML*, 2017.

[85] D. A. Hudson and C. D. Manning, *Gqa: A new dataset for real-world visual reasoning and compositional question answering*, in *CVPR*, 2019.

[86] L. Dong and M. Lapata, *Coarse-to-fine decoding for neural semantic parsing*, in *ACL*, 2018.

[87] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, in *NeurIPS*, 2015.

[88] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, *Bottom-up and top-down attention for image captioning and visual question answering*, in *CVPR*, 2018.

[89] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, *Attention is all you need*, in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[90] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. Tenenbaum, *Neural-symbolic vqa: Disentangling reasoning from vision and language understanding*, in *NeurIPS*, 2018.

[91] Z. Yu, J. Yu, Y. Cui, D. Tao, and Q. Tian, *Deep modular co-attention networks for visual question answering*, in *CVPR*, 2019.

[92] D. A. Hudson and C. D. Manning, *Learning by abstraction: The neural state machine*, in *NeurIPS*, 2019.

[93] R. Vedantam, K. Desai, S. Lee, M. Rohrbach, D. Batra, and D. Parikh, *Probabilistic neural-symbolic models for interpretable visual question answering*, in *ICML*, 2019.

[94] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, *A simple neural network module for relational reasoning*, in *NeurIPS*, 2017.

[95] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, *Film: Visual reasoning with a general conditioning layer*, in *AAAI*, 2018.

[96] Z. Yu, J. Yu, J. Fan, and D. Tao, *Multi-modal factorized bilinear pooling with co-attention learning for visual question answering*, in *ICCV*, 2017.

[97] J.-H. Kim, J. Jun, and B.-T. Zhang, *Bilinear attention networks*, in *NeurIPS*, 2018.

[98] D.-K. Nguyen and T. Okatani, *Improved fusion of visual and language representations by dense symmetric co-attention for visual question answering*, in *CVPR*, 2018.

[99] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh, *Making the v in vqa matter: Elevating the role of image understanding in visual question answering*, in *CVPR*, 2017.

[100] H. Tan and M. Bansal, *Lxmert: Learning cross-modality encoder representations from transformers*, in *EMNLP*, 2019.

[101] J. Lu, D. Batra, D. Parikh, and S. Lee, *Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks*, in *NeurIPS*, 2019.

[102] W. Su, X. Zhu, Y. Cao, B. Li, L. Lu, F. Wei, and J. Dai, *Vl-bert: Pre-training of generic visual-linguistic representations*, *arXiv preprint arXiv:1908.08530* (2019).

[103] Y.-C. Chen, L. Li, L. Yu, A. E. Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu, *Uniter: Learning universal image-text representations*, *arXiv preprint arXiv:1909.11740* (2019).

[104] Z. Gan, Y.-C. Chen, L. Li, C. Zhu, Y. Cheng, and J. Liu, *Large-scale adversarial training for vision-and-language representation learning*, *arXiv preprint arXiv:2006.06195* (2020).

[105] A. Suhr, S. Zhou, A. Zhang, I. Zhang, H. Bai, and Y. Artzi, *A corpus for reasoning about natural language grounded in photographs*, *ACL* (2019).

[106] R. Zellers, Y. Bisk, A. Farhadi, and Y. Choi, *From recognition to cognition: Visual commonsense reasoning*, in *CVPR*, 2019.

[107] C. Zhu, Y. Zhao, S. Huang, K. Tu, and Y. Ma, *Structured attentions for visual question answering*, in *ICCV*, 2017.

[108] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei, *Visual7w: Grounded question answering in images*, in *CVPR*, 2016.

[109] Y. Zhou, R. Ji, J. Su, Y. Wu, and Y. Wu, *More than an answer: Neural pivot network for visual qestion answering*, in *ACMMM*, 2017.

[110] J.-H. Kim, K.-W. On, W. Lim, J. Kim, J.-W. Ha, and B.-T. Zhang, *Hadamard product for low-rank bilinear pooling*, in *ICLR*, 2016.

[111] K. Kafle, B. Price, S. Cohen, and C. Kanan, *Dvqa: Understanding data visualizations via question answering*, in *CVPR*, 2018.

[112] L. Li, Z. Gan, Y. Cheng, and J. Liu, *Relation-aware graph attention network for visual question answering*, *arXiv preprint arXiv:1903.12314* (2019).

[113] R. Hu, A. Rohrbach, T. Darrell, and K. Saenko, *Language-conditioned graph networks for relational reasoning*, in *ICCV*, 2019.

[114] Z. Yang, X. He, J. Gao, L. Deng, and A. Smola, *Stacked attention networks for image question answering*, in *CVPR*, 2016.

[115] R. Cadene, H. Ben-Younes, M. Cord, and N. Thome, *Murel: Multimodal relational reasoning for visual question answering*, in *CVPR*, 2019.

[116] C. Wu, Y. Zhou, G. Li, N. Duan, D. Tang, and X. Wang, *Deep reason: A strong baseline for real-world visual reasoning*, *arXiv preprint arXiv:1905.10226* (2019).

[117] D. Guo, C. Xu, and D. Tao, *Graph reasoning networks for visual question answering*, *arXiv preprint arXiv:1907.09815* (2019).

[118] P. Liang, M. I. Jordan, and D. Klein, *Learning semantic correspondences with less supervision*, in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pp. 91–99, Association for Computational Linguistics, 2009.

[119] T. Liu, K. Wang, L. Sha, B. Chang, and Z. Sui, *Table-to-text generation by structure-aware seq2seq learning*, in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[120] O. Dušek, J. Novikova, and V. Rieser, *Evaluating the state-of-the-art of end-to-end natural language generation: The E2E NLG Challenge*, *arXiv preprint arXiv:1901.11528* (Jan., 2019).

[121] S. Wiseman, S. Shieber, and A. Rush, *Challenges in data-to-document generation*, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2253–2263, 2017.

[122] A. See, P. J. Liu, and C. 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)*, pp. 1073–1083, 2017.

[123] J. Gu, Z. Lu, H. Li, and V. O. Li, *Incorporating copying mechanism in sequence-to-sequence learning*, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1631–1640, 2016.

[124] R. Puduppully, L. Dong, and M. Lapata, *Data-to-text generation with content selection and planning*, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 6908–6915, 2019.

[125] R. Puduppully, L. Dong, and M. Lapata, *Data-to-text generation with entity modeling*, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 2023–2035, Association for Computational Linguistics, July, 2019.

[126] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, *A large annotated corpus for learning natural language inference*, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642, 2015.

[127] A. Talmor, J. Herzig, N. Lourie, and J. Berant, *Commonsenseqa: A question answering challenge targeting commonsense knowledge*, 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)*, pp. 4149–4158, 2019.

[128] A. Rohrbach, L. A. Hendricks, K. Burns, T. Darrell, and K. Saenko, *Object hallucination in image captioning*, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4035–4045, 2018.

[129] B. Dhingra, M. Faruqui, A. Parikh, M.-W. Chang, D. Das, and W. Cohen, *Handling divergent reference texts when evaluating table-to-text generation*, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4884–4895, 2019.

[130] N. Ford, D. Duckworth, M. Norouzi, and G. Dahl, *The importance of generation order in language modeling*, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2942–2946, 2018.

[131] S. Welleck, K. Brantley, H. D. Iii, and K. Cho, *Non-monotonic sequential text generation*, in *International Conference on Machine Learning*, pp. 6716–6726, 2019.

[132] D. L. Chen and R. J. Mooney, *Learning to sportscast: a test of grounded language acquisition*, in *Proceedings of the 25th international conference on Machine learning*, pp. 128–135, ACM, 2008.

[133] R. Lebret, D. Grangier, and M. Auli, *Neural text generation from structured data with application to the biography domain*, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1203–1213, 2016.

[134] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, *A neural probabilistic language model*, *Journal of machine learning research* **3** (2003), no. Feb 1137–1155.

[135] S. Bird, *Nltk: the natural language toolkit*, in *Proceedings of the COLING/ACL on Interactive presentation sessions*, pp. 69–72, Association for Computational Linguistics, 2006.

[136] P. Liang, M. I. Jordan, and D. Klein, *Learning dependency-based compositional semantics*, *Computational Linguistics* **39** (2013), no. 2 389–446.

[137] W. Kryściński, B. McCann, C. Xiong, and R. Socher, *Evaluating the factual consistency of abstractive text summarization*, *arXiv preprint arXiv:1910.12840* (2019).

[138] I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, *arXiv preprint arXiv:1412.6572* (2014).

[139] A. Kannan and O. Vinyals, *Adversarial evaluation of dialogue models*, *arXiv preprint arXiv:1701.08198* (2017).

[140] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, *Neural computation* **9** (1997), no. 8 1735–1780.

[141] A. Liu, J. Du, and V. Stoyanov, *Knowledge-augmented language model and its application to unsupervised named-entity recognition*, 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)*, pp. 1142–1150, 2019.

[142] N. Zhang, S. Deng, Z. Sun, J. Chen, W. Zhang, and H. Chen, *Relation adversarial network for low resource knowledgegraph completion*, *arXiv preprint arXiv:1911.03091* (2019).

[143] S. Ramakrishnan, A. Agrawal, and S. Lee, *Overcoming language priors in visual question answering with adversarial regularization*, in *Advances in Neural Information Processing Systems*, pp. 1541–1551, 2018.

[144] T.-H. Wen, M. Gasic, N. Mrkšić, P.-H. Su, D. Vandyke, and S. Young, *Semantically conditioned lstm-based natural language generation for spoken dialogue systems*, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1711–1721, 2015.

[145] S. Sharma, J. He, K. Suleman, H. Schulz, and P. Bachman, *Natural language generation in dialogue using lexicalized and delexicalized data*, *ICLR Workshop* (2017).

[146] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in *ICLR*, 2014.

[147] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, *Huggingface's transformers: State-of-the-art natural language processing*, *ArXiv* **abs/1910.03771** (2019).

[148] K. Kukich, *Design of a knowledge-based report generator*, in *Proceedings of the 21st annual meeting on Association for Computational Linguistics*, pp. 145–150, Association for Computational Linguistics, 1983.

[149] P. Holmes-Higgin, *Text generation—using discourse strategies and focus constraints to generate natural language text by kathleen r. mckeown, cambridge university press, 1992, pp 246,£ 13.95, isbn 0-521-43802-0.*, *The Knowledge Engineering Review* **9** (1994), no. 4 421–422.

[150] S. Wiseman, S. Shieber, and A. Rush, *Learning neural templates for text generation*, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3174–3187, 2018.

[151] J. Gu, Q. Liu, and K. Cho, *Insertion-based decoding with automatically inferred generation order*, *TACL* (2019).

[152] E. Mansimov, A. Wang, and K. Cho, *A generalized framework of sequence generation with application to undirected sequence models*, *arXiv preprint arXiv:1905.12790* (2019).

[153] B. Goodrich, V. Rao, P. J. Liu, and M. Saleh, *Assessing the factual accuracy of generated text*, in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 166–175, ACM, 2019.

[154] É. Colin and C. Gardent, *Generating text from anonymised structures*, in *Proceedings of the 12th International Conference on Natural Language Generation*, pp. 112–117, 2019.

[155] Z. Dai, Z. Yang, Y. Yang, W. W. Cohen, J. Carbonell, Q. V. Le, and R. Salakhutdinov, *Transformer-xl: Attentive language models beyond a fixed-length context*, *ACL* (2019).

[156] D. Vrandečić and M. Krötzsch, *Wikidata: a free collaborative knowledgebase*, *Communications of the ACM* **57** (2014), no. 10 78–85.

[157] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, *Distant supervision for relation extraction without labeled data*, in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pp. 1003–1011, Association for Computational Linguistics, 2009.

[158] A. Shimorina and C. Gardent, *Handling rare items in data-to-text generation*, in *Proceedings of the 11th International Conference on Natural Language Generation*, pp. 360–370, Association for Computational Linguistics, 2018.

[159] E. Loper and S. Bird, *Nltk: The natural language toolkit*, in *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pp. 63–70, 2002.

[160] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, *Gated graph sequence neural networks*, *ICLR* (2016).

[161] T. N. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, *ICLR* (2017).

[162] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, *Graph attention networks*, *ICLR* (2018).

[163] Z. Chen, H. Eavani, W. Chen, Y. Liu, and W. Y. Wang, *Few-shot nlg with pre-trained language model*, *ACL* (2020).

[164] M.-T. Luong, H. Pham, and C. D. Manning, *Effective approaches to attention-based neural machine translation*, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, 2015.

[165] D. Marcheggiani and L. Perez-Beltrachini, *Deep graph convolutional encoders for structured data to text generation*, in *Proceedings of the 11th International Conference on Natural Language Generation*, pp. 1–9, 2018.

[166] O. Dušek and F. Jurcicek, *Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings*, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 45–51, 2016.

[167] J. Juraska, P. Karagiannis, K. Bowden, and M. Walker, *A deep ensemble model with slot alignment for sequence-to-sequence natural language generation*, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 152–162, 2018.

[168] H. Elder, S. Gehrmann, A. O'Connor, and Q. Liu, *E2e nlg challenge submission: Towards controllable generation of diverse natural language*, in *Proceedings of the 11th International Conference on Natural Language Generation*, pp. 457–462, 2018.

[169] J. Bao, D. Tang, N. Duan, Z. Yan, Y. Lv, M. Zhou, and T. Zhao, *Table-to-text: Describing table region with natural language*, in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[170] L. Sha, L. Mou, T. Liu, P. Poupart, S. Li, B. Chang, and Z. Sui, *Order-planning neural text generation from structured data*, in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[171] S. Chen, J. Wang, X. Feng, F. Jiang, B. Qin, and C.-Y. Lin, *Enhancing neural data-to-text generation models with external background knowledge*, 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)*, pp. 3013–3023, 2019.

[172] T. Liu, F. Luo, Q. Xia, S. Ma, B. Chang, and Z. Sui, *Hierarchical encoder with auxiliary supervision for neural table-to-text generation: Learning better representation for tables*, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 6786–6793, 2019.

[173] S. Ma, P. Yang, T. Liu, P. Li, J. Zhou, and X. Sun, *Key fact as pivot: A two-stage model for low resource table-to-text generation*, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2047–2057, 2019.

[174] S. Wiseman, S. Shieber, and A. Rush, *Challenges in data-to-document generation*, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (Copenhagen, Denmark), pp. 2253–2263, Association for Computational Linguistics, Sept., 2017.

[175] S. Ahn, H. Choi, T. Pärnamaa, and Y. Bengio, *A neural knowledge language model*, *arXiv preprint arXiv:1608.00318* (2016).

[176] H. Hayashi, Z. Hu, C. Xiong, and G. Neubig, *Latent relation language models*, *Thirty-Fourth AAAI Conference on Artificial Intelligence* (2020).

[177] R. Logan, N. F. Liu, M. E. Peters, M. Gardner, and S. Singh, *Barack's wife hillary: Using knowledge graphs for fact-aware language modeling*, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5962–5971, 2019.

[178] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, *Improving language understanding by generative pre-training*, *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf* (2018).

[179] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, *Xlnet: Generalized autoregressive pretraining for language understanding*, in *Advances in neural information processing systems*, pp. 5754–5764, 2019.

[180] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher, *Ctrl: A conditional transformer language model for controllable generation*, *arXiv preprint arXiv:1909.05858* (2019).

[181] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, *Albert: A lite bert for self-supervised learning of language representations*, *ICLR* (2020).

[182] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, *Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*, *arXiv preprint arXiv:1910.13461* (2019).

[183] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, *Exploring the limits of transfer learning with a unified text-to-text transformer*, *arXiv preprint arXiv:1910.10683* (2019).